

AD-A216 192

④

Technical Report 1286
October 1989

A Fuzzy Logic Decision Support Tool

R. W. Larsen
R. A. Dillard

DTIC
ELECTE
DEC 27 1989
S B D

Approved for public release; distribution is unlimited.

89 12 26 140

NAVAL OCEAN SYSTEMS CENTER

San Diego, California 92152-5000

J. D. FONTANA, CAPT, USN
Commander

R. M. HILLYER
Technical Director

ADMINISTRATIVE INFORMATION

This work was performed under Program Element 0602936N for the Office of Naval Research, Independent Exploratory Development Programs, Arlington, VA 22217.

Released by
D. C. Eddington, Head
Decision Support and AI
Systems Branch

Under authority of
W. T. Rasmussen, Head
Command Support Technology
Division

ACKNOWLEDGMENT

The authors would like to acknowledge Darlene Teotico and Allen Shum who implemented the fuzzy logic computer program and Nadja O'Hagen who suggested the topic.

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED			1b. RESTRICTIVE MARKINGS	
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release: distribution is unlimited.	
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE				
4. PERFORMING ORGANIZATION REPORT NUMBER(S) NOSC TR 1286			5. MONITORING ORGANIZATION REPORT NUMBER(S)	
6a. NAME OF PERFORMING ORGANIZATION Naval Ocean Systems Center	6b. OFFICE SYMBOL (if applicable) Code 444	7a. NAME OF MONITORING ORGANIZATION		
6c. ADDRESS (City, State and ZIP Code) San Diego, CA 92152-5000		7b. ADDRESS (City, State and ZIP Code)		
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Office of Naval Research	8b. OFFICE SYMBOL (if applicable) ONR	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER		
8c. ADDRESS (City, State and ZIP Code) Independent Exploratory Development Programs OCNR-20T Arlington, VA 22217		10. SOURCE OF FUNDING NUMBERS		
		PROGRAM ELEMENT NO. 0602936N	PROJECT NO. ZE66	AGENCY ACCESSION NO. DN308 060
11. TITLE (include Security Classification) A FUZZY LOGIC DECISION SUPPORT TOOL				
12. PERSONAL AUTHOR(S) R. W. Larsen and R. A. Dillard				
13a. TYPE OF REPORT Final	13b. TIME COVERED FROM Oct 88 TO Mar 89	14. DATE OF REPORT (Year, Month, Day) Oct 1989	15. PAGE COUNT 121	
16. SUPPLEMENTARY NOTATION				
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)	
FIELD	GROUP	SUB-GROUP		
			fuzzy logic	
			decision tool	
			artificial intelligence	
19. ABSTRACT (Continue on reverse if necessary and identify by block number) This report describes a fuzzy logic decision tool that has broad application to command and control systems. The report presents an algorithm for solving multiple-objective decision problems when the objectives have differing degrees of importance.				
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT <input type="checkbox"/> UNCLASSIFIED/UNLIMITED <input checked="" type="checkbox"/> SAME AS RPT <input type="checkbox"/> DTIC USERS			21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a. NAME OF RESPONSIBLE PERSON R. W. Larsen		22b. TELEPHONE (include Area Code) (619) 553-4142	22c. OFFICE SYMBOL Code 444	

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

CONTENTS

SUMMARY	iii
INTRODUCTION	1
Background	1
Description	1
FUZZY SET THEORY	3
Overview	3
Fuzzy versus Probability Theory	3
Fuzzy Logic	4
Fuzzy Operations	5
Membership Operation	5
Complement Operation	6
Intersection Operation	6
Union Operation	7
Exponentiation Operation	7
BASIS FOR ALGORITHM DEVELOPMENT	7
Impartial Fuzzy Decisions	7
Partial Fuzzy Decisions	9
Pair-Wise Objective Evaluation	10
An Example	12
Observations	15
Rank Order Objective Evaluation	16
An Example	17
Observations	20
Utility Functions	21
An Example	22
Explanation Facility	23
Explanation for the Min-Max Rank Method	23
Explanation for the Min-Max Pair-Wise Method	25
Explanation for the Weighted-Sum Rank Method	28
Explanation for the Weighted-Sum Pair-Wise Method	29
Observations	30
The Eigenvector Method of Finding Weights	30
Approximation to the Maximum Eigenvalue and Normalized Eigenvector	30
Notes on the Eigenvector Method	33
Methods for Finding the Worst Inconsistencies	33
Column Method	34
Triple Method	35



Dist

A-1

Special

or	<input checked="" type="checkbox"/>
	<input type="checkbox"/>
	<input type="checkbox"/>
n/	
ty Codes	
and/or	

CONTENTS (contd)

Comparison to Dempster's Rule	38
Dempster's Rule	38
Examples of Dempster's Rule	40
Problem Domain Differences	41
Dempster's Rule Applied to Weighted Assignments	42
Weighting Assignments for Use in Dempster's Rule — A Summary	44
Conversion of Fuzzy Subsets into Mass Assignments	44
Examples of Conversion	45
Experimental Comparisons	46
Applications of the Fuzzy Logic Decision Support Tool	46
A Plan Selection Aid for Theater Strike Mission Planning	47
Plan Generator and Evaluator for the Air Strike Planning Advisor	53
References	55
Appendices:	
A. Programmed Examples	A-1
B. Functional Diagrams	B-1
C. Function Descriptions	C-1
D. Program Listing for Min-Max Pair-Wise Method (FranzLISP)	D-1
E. Program Listing for Min-Max Rank Method (FranzLISP)	E-1

TABLES

1. Level-of-importance definitions	11
2. A linguistic likelihood scale	13
3. ASPA Defense Suppression: plan evaluation criteria	54

FIGURES

1. Candidate missions	49
2. PSA Step One	50
3. PSA Step Two	51
4. PSA Step Three	52
5. Deception Plan Generator	54

SUMMARY

This report describes a fuzzy logic decision tool developed at NOSC under Independent Exploratory Development funding. The tool implements alternate versions of an algorithm based on results published by Zadeh, Bellman, Saaty, and Yager. The tool has broad application to command and control systems since decision making in a military context usually involves imprecise knowledge and subjective goals.

Decision making is defined as an act of rational choice. In particular, given a set of possible choices for a given situation, the optimal choice is the one that meets a particular set of objectives *best*. Best implies that a utility function over objectives can be optimized.

An important enhancement of this model uses fuzzy sets to represent uncertain objectives and incomplete information about a situation (Zadeh & Bellman, 1970). The ability to represent multiple-objective decision problems involving vague or fuzzy objectives is, in fact, one of the more useful applications of fuzzy set theory. This report presents an algorithm for solving multiple-objective decision problems when the objectives have differing degrees of importance. *Commonly used, from multiple sources, some data included.*

The input to the algorithm is (1) a set of choices, (2) a set of objectives, (3) the degree to which each choice satisfies each objective, and (4) a comparison of the importance of the objectives. Two methods have been developed for comparing the importance of the objectives. These are the rank order method (Yager, 1981) and the pair-wise comparison method (Saaty, 1977). The rank order method is simple to use and solves small problems quickly and efficiently. The pair-wise comparison method is more effective for large objective sets because it allows inconsistencies in user preferences to be evaluated.

Two utility functions to be optimized over the objectives have been implemented. The first, the *min-max* method, corresponds to worst-case analysis, while the second method, *weighted-sum*, corresponds to average-case analysis. The appropriate method to use depends on the specific decision problem, particularly the expected impact of a wrong decision.

Another important feature that has been implemented is a decision explanation. The explanation facility assumes that the user expects a particular choice. The candidate choice that scores best for the two most important objectives is considered to be the expected choice of the user. If this is not the choice selected by the algorithm, then the difference between the expected choice and the actual choice is explained to the user.

This fuzzy logic decision support tool has been used to aid the knowledge acquisition process for expert system development at NOSC. It has also been integrated into two advanced mission planning aids, *Theater Strike Mission Planning* and the *Air Strike Planning Advisor* to support the evaluation and selection of alternative plans and their component parts.

INTRODUCTION

BACKGROUND

Humans deal naturally with subjective or fuzzy information. Typically, we speak of tall men in easily understood conversation, although in actual fact, we cannot precisely define what a tall man is. Likewise, military command and control, by its nature, deals extensively with imprecise knowledge and subjective goals. The state of a battlefield situation is usually not well known. There is never enough information or time to completely analyze a situation in order to make a decision. Yet humans tend to perform reasonably well under such circumstances, arriving at good decisions in spite of ambiguity and confusion.

On the other hand, the volume of information and the pace of modern warfare operations preclude the time-consuming and labor-intensive processes of the past. In addition, despite their best efforts, human feelings and preferences remain inconsistent and intransitive, often leading to judgmental errors. The battlefield of the future has an overriding need for computerized information and decision support systems that support rapid, reliable, and effective assimilation of timely information for planning, decisions, and command action.

DESCRIPTION

The fuzzy logic decision tool discussed here provides a mechanism to support systematic decision making in knowledge-based planning systems and decision aids. It has been applied to the process of knowledge acquisition in order to structure and build knowledge bases as well as construct user interfaces that support subjective decision making.

The fuzzy logic decision support tool provides (1) four different methods to obtain a recommended decision (i.e., Min-Max Rank, Min-Max Pair-Wise, Weighted-Sum Rank, and Weighted-Sum Pair-Wise Methods) and (2) brief descriptions for each of the above methods. Once a method is selected, the user is prompted for the numbers of, and labels to be assigned to, alternatives and criteria. Methods 1 and 3 (the Rank methods) require the user to rank the criteria in order of importance. Methods 2 and 4 (the Pair-Wise methods) do not require that the user be able to rank criteria. Instead, he/she only must be able to compare the importance of criteria one against the other in a pair-wise fashion. Once the user inputs the required data, the best choice decision is produced as well as a ranked decision list. At this point the user can select to either obtain an explanation for the recommended decision, review his/her inputs, return to the top menu, or quit. Upon request, explanations are provided in natural language format. If inconsistencies exist in the inputs, a list of significant inconsistencies is provided followed by a recommended change to minimize the inconsistency. The user can select the recommended change and obtain a new decision, start over, or quit.

The next sections in this report describe the decision tool, the derivation of all the formulas and concepts used in the development of the decision tool, and two prototype applications demonstrated with this tool. Appendix A provides a comprehensive set of programmed examples that were generated to coincide with examples in

the text. The VAX computer program, written in FranzLISP* (UNIX environment), that generated these examples is documented in appendices B and C. Appendices D and E contain the fuzzy logic decision tool's most useful components. These listings are stand-alone source code for the Min-Max Pair-Wise Method and the Min-Max Rank Method, respectively, which are core algorithms for the complete decision tool. The Weighted-Sum Method, the explanation facility, the inconsistency analyzer, and the complete user interface are not included in these shortened versions of the source code.

*Note: CommonLISP and InterLISP versions are also available.

FUZZY SET THEORY

OVERVIEW

Zadeh (1965) formulated the initial statement of fuzzy set theory. Since then, this mathematical discipline has gone through substantial theoretical development. Correspondingly, there has been a proliferation of applications of this basic mathematical framework to a variety of fields.

Ordinary set theory principles underlie modern mathematics. Fundamental to basic set theory is the notion that an item is either a member or is not a member of a set. However, the fact is that in the real world, membership in a set is not always so well-defined. Fuzzy set theory is based on recognition that certain sets have imprecise boundaries. Fuzzy sets are those ill-specified and nondistinct collections of objects with unsharp boundaries in that the transition from membership to nonmembership in a subset of a reference set is gradual rather than abrupt.

Typically, we speak of tall men or expensive homes. Membership in such sets or classes of objects is not characterized by either/or, but are sets in that membership can be adequately considered in terms of degrees. A fuzzy set is characterized by a membership function, defined as a real number in the interval $[0,1]$. For example, a membership measure $\mu_A(x_i) = 0.8$ suggests that member x_i of set X is a member of the fuzzy set A to a degree 0.8 on a scale where zero is no membership and one is complete membership. It should be clear that fuzzy set theory can be reduced to ordinary set theory by constraining membership to the extremes of the range 0 to 1.

Just as traditional set theory operations can be precisely defined, fuzzy set operations can be precisely defined. For example, the complement A' of a fuzzy set A is

$$\mu_{A'}(x_i) = 1 - \mu_A(x_i) \text{ for all } x_i \text{ in } X,$$

or, in abbreviated form,

$$\mu_{A'}(X) = 1 - \mu_A(X).$$

The intersection of fuzzy set A and fuzzy set B is fuzzy set $C = A \cap B$ where

$$\mu_{C(X)} = \min \{ \mu_A(X), \mu_B(X) \},$$

and the union of fuzzy set A and fuzzy set B is the fuzzy set $C = A \cup B$ where

$$\mu_{C(X)} = \max \{ \mu_A(X), \mu_B(X) \}.$$

As with ordinary set theory, fundamental notions such as distributive and associative properties apply to fuzzy sets. In fact, fuzzy set theory has been developed to the point that formal manipulation is fully defined as with ordinary set theory.

FUZZY VERSUS PROBABILITY THEORY

Before continuing, a fundamental clarification should be made that concerns how the imprecision of fuzzy set theory or possibility theory differs from the

imprecision dealt with by probability theory. Basically, the difference is that probability theory deals with randomness of future events, whereas possibility theory deals with the imprecision of current or past events. Randomness deals with the uncertainty regarding the occurrence or nonoccurrence of some event, while the imprecision of fuzzy sets deals with the membership or nonmembership of an object in a set with imprecise boundaries.

A typical probabilistic statement is "There is a 10-percent chance that the next person to enter the room will be over 6 feet tall." A typical possibilistic statement is "John is tall." The probabilistic statement refers to a precise set of people over 6 feet tall. The imprecision in this case has to do with the event relating to the next person in the room. The fuzzy statement is not imprecise about the event in question; it is "John." The imprecision here has to do with the vagueness of the concept of "tall" itself.

There is the possibility of combining these two concepts; for example, "There is a 10-percent chance that the next person in the room will be tall." Such theory is a topic of considerable interest. However, the point here is that the fuzzy concept deals with a dimension of uncertainty that is generally quite distinct from that of probability theory.

FUZZY LOGIC

Logic is the science of the normative formal principles of reasoning. In this sense, fuzzy logic is concerned with the formal principles of approximate reasoning, with precise reasoning viewed as a limiting case.

In more specific terms, what is central about fuzzy logic is that, unlike classical logical systems, it aims at modeling the imprecise modes of reasoning that play an essential role in the remarkable human ability to make rational decisions in an environment of uncertainty and imprecision. This ability depends, in turn, on our ability to infer an approximate answer to a question based on a store of knowledge that is inexact, incomplete, or not totally reliable. For example, "John is much taller than most of his close friends. How tall is John?"

There are two main reasons why classical logical systems cannot cope with problems of this type. First, they do not provide a system for representing the meaning of propositions expressed in a natural language when the meaning is precise, and second, in those cases in which the meaning can be represented symbolically in a meaning representation language (e.g., a semantic network or a conceptual dependency graph) there is no mechanism for inference. The expressive power of fuzzy logic derives from the fact that it contains as special cases not only the classical two-valued and multivalued logical systems but also probability theory and probabilistic logic.

During the past several years, fuzzy logic has found numerous applications in fields ranging from finance to earthquake engineering. The most important and striking application has been the realm of fuzzy-logic-based process control. Among the well-publicized applications are automatic train operation, vehicle control, robot control, speech recognition, universal controllers, and stabilization control.

Although current applications of fuzzy logic are in the form of software algorithms, it is clear that it would be cheaper and more effective to use fuzzy logic chips

and, eventually, fuzzy computers. Such technology has been demonstrated in the laboratory and will be available for commercial use in the near future. These developments will lead to an expanded use of fuzzy logic not only in industrial process control but, more generally, in knowledge-based systems in which the deduction of an answer to a query requires the inference machinery of fuzzy logic. Of particular interest here is the application of process control models to the *softer* social, economic, and management problems. The optimization of fuzzy goals in the context of fuzzy constraints is a key form of management control that is at the core of the command and control decision process.

A crucial aspect of decision making involves the handling of the information available about the uncertainty in a decision. In many instances, the information concerning the uncertainty is not strong enough to make probabilistic statements about the situation. In other situations, particularly one of a kind, the information on uncertainty does not have the character of a probability. As before, the statement that "John is tall" tells us something about John's height. It gives us information about his size or physical appearance with some uncertainty, though not of a probability kind. (Yager, 1978) has shown that possibilistic information is closely related to an ordinal theory of uncertainty, while probability involves a cardinal theory of uncertainty.

FUZZY OPERATIONS

Here are some fuzzy set operations of interest.

Membership Operation

Suppose we have a set of alternatives, such as a set of countries to which one might travel. Let the set of decision alternatives be denoted by $X = \{x_1, \dots, x_n\}$, which for our specific example might look like

$$X = \{\text{Brazil, India, China, Mexico}\}$$

representing our potential destination countries of choice. A fuzzy subset A of X is characterized by a membership function $\mu(X)$ that associates with every member of X a number in the interval $[0,1]$ that indicates the grade of membership of X_i in A . Suppose, for our example, A is a fuzzy subset defined as

$$A = \text{the country is near the U.S.}$$

Then,

$$\{\mu A(x_i)/x_i\} = \{0.6/\text{Brazil}, 0.001/\text{India}, 0.9/\text{Cuba}, 1.0/\text{Mexico}\}.$$

Although the above notation is used by some authors, it is also common to simply write

$$A = \{0.6, 0.001, 0.9, 1.0\}$$

where $\{\mu A(x_i)\}$ is actually meant where A is used above. Since there seems to be no accepted standard, the simpler notation will be used for readability in this report.

What we have in this case is $\mu A(\text{Brazil}) = 0.6$, $\mu A(\text{India}) = 0.001$, $\mu A(\text{Cuba}) = 0.9$, and $\mu A(\text{Mexico}) = 1.0$. The point is to note the increasing membership grades as a country distance from the U.S. decreases. As mentioned before, the concept of a fuzzy set includes that of a classical set as a special case using a step function for the membership function. For example,

$$\begin{aligned} X &= \{1, 2, 3, 4, 5\}, \\ A &= \text{the set greater than 2}, \\ &= \{0, 0, 1, 1, 1\}. \end{aligned}$$

Note that 1 and 2 do not belong to the set A, so their membership grade is 0, while 3, 4, and 5 have membership grades of 1 since they absolutely belong to the set A in exactly the classical set sense.

Complement Operation

In the preceding example, let

$$A' = \text{the country is not near the U.S.}$$

Then, $\mu A' = 1 - \mu A$, so we have

$$\begin{aligned} A' &= \{1 - 0.6, 1 - 0.001, 1 - 0.9, 1 - 1.0\} \\ &= \{0.4, 0.999, 0.1, 0\}. \end{aligned}$$

Intersection Operation

If A and B are two fuzzy sets defined on X, their intersection, a fuzzy set C, is denoted by $C = A \cap B$ and has a membership function that is the minimum of the respective elements of the membership functions of A and B for all x_i in X. This corresponds to the logical "and" operation. As an example, let X be a set of potential weapons $\{x_1, x_2, x_3\}$. Let A be defined as the set of inexpensive weapons. Then relative to X we might have

$$A = \{0.6, 0.5, 0.8\}.$$

That is, this fuzzy set indicates how well each of the weapons satisfies the objective of being cheap. Let another fuzzy set B be defined as

$$\begin{aligned} B &= \text{the set of weapons that are effective} \\ &= \{0.7, 0.3, 0.9\}. \end{aligned}$$

The fuzzy subset C of both *inexpensive and effective* weapons would be

$$C = A \cap B = \{0.6, 0.3, 0.8\}$$

with the membership function $\mu C(X)$ equal to the minimum of the respective fuzzy subsets A and B describing how well a particular weapon alternative satisfies the condition of being inexpensive and effective. For example, $\mu C(X_1) = \min \{0.6, 0.7\} = 0.6$, means weapon x_1 has a 0.6 chance of being both inexpensive and effective.

Union Operation

On the other hand, if A and B are two fuzzy sets defined on X, their union, a fuzzy set C, is denoted by $C = A \cup B$ and has a membership function that is the maximum of the respective elements of the membership functions of A and B for all x_i in X. This corresponds to the logical "or" operation. For this example, the fuzzy subset C of either *inexpensive* or *effective* weapons would be

$$C = A \cup B = \{0.7, 0.5, 0.9\}$$

with the membership function $\mu_C(X)$ equal to the maximum of the respective fuzzy subsets A and B describing how well a particular weapon alternative satisfies either the condition of being inexpensive or effective.

Exponentiation Operation

Let A be a fuzzy subset over X, and let $\alpha \geq 0$ be a scalar. The operation of exponentiation, denoted by $B = A^\alpha$ is defined as a fuzzy set over X with membership function

$$\mu_B(X) = \mu_A(X)^\alpha$$

Using the same set A of inexpensive weapons above, for $\alpha = 2$, we get the set of *very* inexpensive (*really* cheap) weapons

$$\begin{aligned} B = A^2 &= \{0.6^2, 0.5^2, 0.8^2\} \\ &= \{0.36, 0.25, 0.64\}. \end{aligned}$$

Zadeh originally suggested using the power of 2 to represent the fuzzy linguistic modifier *very*. Note that when $\alpha > 1$, the effect of raising A to the power α is to reduce the grade of membership for all the x_i 's, but in such a manner that those that have large membership values are reduced much less than those that are small. This has the effect of making the requirements more stringent: the bigger α , the more stringent. For $\alpha < 1$, the opposite is true.

BASIS FOR ALGORITHM DEVELOPMENT

IMPARTIAL FUZZY DECISIONS

One type of decision consists of a situation in which a group of objectives is given in terms of requirements that the selected solution should have, for example, the desired weapon should be inexpensive, effective, reliable, available, etc. The decision process is to select the candidate that best satisfies all these objectives. As Bellman and Zadeh suggest in their 1970 paper that uses the rule of implied conjunction, these objectives are stated as

$$C_1 \text{ and } C_2 \text{ and } C_3 \dots$$

If we associate with each objective a fuzzy subset over the set of potential candidate alternatives, then, in terms of fuzzy subsets, our decision D becomes

$$D = C_1 \cap C_2 \cap C_3 \dots,$$

where C_i is a fuzzy subset of the decision alternatives whose membership function indicates how well each of the candidates satisfies that objective. In addition, the decision D is also a fuzzy subset over the alternatives whose membership function $\mu_D(X)$ indicates how well each of the alternatives satisfies the set of objectives. We then select the alternative that has the highest grade of membership in D as the best alternative, because it satisfies the set of objectives with the highest value. This decision rule is referred to as impartial because each objective is treated equally.

As an example, suppose that the set of possible decision alternatives is

$$X = \{\text{merchant ship, surface warship, submarine}\}.$$

Suppose that a particular acoustic measurement is correlated with the above ship classes as follows:

$$A = \{0.5, 0.7, 0.8\}.$$

That is, this fuzzy set indicates how an observed acoustic signature fits three potential classification hypotheses. The fuzzy membership data could be the result of an operator's linguistic description of his/her interpretation of observed data, or it could be the result of matches returned from a fuzzy query into an acoustic database. In the absence of other data, our current *best* guess at the classification of the acoustic contact would be a submarine. Suppose, however, that an additional classification based on an electronic emission were also available as follows:

$$B = \{0.8, 0.6, 0.7\}.$$

This indicates the best guess at a contact's identity is merchant ship. Using the concept of intersection introduced above, we can create a new fuzzy set of a joint acoustic and electronic contact. This fuzzy subset would be given by

$$C = A \cap B = \{0.5, 0.6, 0.7\},$$

that is, the membership function $\mu_C(X)$ indicates how well a contact fits each of the hypothesized ship categories combining both acoustic and electronic classification estimates. Considering both types of information, the submarine still is most likely.

The best decision is submarine because it corresponds to the decision alternative with the highest value, namely 0.7. Note that the procedure presented so far is under the condition that all information is equally important, which is usually not the case. But, before proceeding, let us make some observations about fuzzy decisions.

Based upon the definition of the intersection operation, we can observe the following:

The decision is made by

1. selecting for each alternative x_i , its minimum membership value in any of the objectives;

2. then selecting as our optimal decision the alternative with the maximum membership in D, the decision set.

In the previous example, the maximum membership grade of 0.7 in the decision set is the minimum of the membership grades for a submarine (i.e., the minimum of the values 0.7 and 0.8). For this reason, this decision rule is also referred to as the max-min method. Note this method should be distinguished from the usual game-theoretic min-max method.

PARTIAL FUZZY DECISIONS

The concept of using the conjunction or intersection of goal, objective, and constraint sets that are subsets of some space of decision alternatives was first formulated by Zadeh and Bellman (1970). The decision is fuzzy if any of the goal, objective, or constraint subsets is fuzzy.

In many cases, the importance of the goal, objective, or constraint sets entering the fuzzy decision process will not be of equal importance. If a particular objective is of great importance, we want to be very unlikely to select any alternative as our solution that has a small membership value in this objective. This can be accomplished by making those alternatives that are low in important objectives have a low membership in D, thereby minimizing the chance of their being selected as the best. From statement (1) of the decision rule above, we can conclude that the membership function for each alternative in D is determined by its lowest membership in all the objectives. Therefore, if we make the grade of membership of the alternatives that are bad in important objectives even lower, they will be less likely to be selected as the optimal alternative.

Recalling the effects of raising a fuzzy set to a power, explained previously, we see that if we assign to each objective a number $\alpha \geq 0$ indicative of its importance, we can obtain the above desired effects. That is, if we associate with each objective an α (the more important the objective, the higher the α) and then consider our decision as

$$D = C_1^{\alpha_1} \cap C_2^{\alpha_2} \cap \dots \cap C_n^{\alpha_n},$$

where

$$\alpha_i \geq 0, i = 1, 2, \dots, n,$$

$$\frac{1}{n} \sum_{i=1}^n \alpha_i = 1$$

we have a situation where alternatives that are weak in important constraints become even less appealing as potential solutions. Furthermore, the linguistic characterization associated with exponentiation operations reinforces this approach in the sense that the more stringent we are enforcing our objective, the more important it is. Thus, for the fuzzy set A of inexpensive weapons, with $\alpha = 2$, we would call A^2 the set of very inexpensive weapons, while $\alpha = 1/2$ would give us the set $A^{1/2}$ of more or less inexpensive weapons.

Therefore, in a sense we have a hierarchical system in that each alternative is first rated on its ability to satisfy each of the objectives and then the objectives are rated as to their importance.

Now let us suppose for our previous example that there exists another intelligence source that can also produce fuzzy membership estimates for the three ship classes as follows:

$$C = \{0.9, 0.5, 0.4\}.$$

Now let us presume that acoustic classification is inferior to this intelligence source, but that electronic classification is superior. In other words, we might want to raise the membership function for acoustic classification to a more stringent requirement, such as $\alpha = 1/2$, whereas electronic classification might have the less stringent requirement of 2. Applying the methodology of Yager and Saaty, the decision set D is formed as follows:

$$D = A^{1/2} \cap B^2 \cap C^1,$$

or

$$D = \{0.5, 0.7, 0.8\}^{1/2} \cap \{0.8, 0.6, 0.7\}^2 \cap \{0.9, 0.5, 0.4\}^1$$

$$\begin{aligned} D &= \{0.70, 0.83, 0.89\} \cap \{0.64, 0.36, 0.49\} \cap \{0.9, 0.5, 0.4\}, \\ &= \{0.64, 0.36, 0.4\}. \end{aligned}$$

Because of its relative importance, the electronic emissions classification *overrides* the other two data sources and favors the decision of merchant ship as the most probable contact. It should be noted that the above example, as well as others in this report, are used for illustrative purposes only and are not intended to represent the real complexity of the actual problem cited.

The next problem is that of obtaining a scale on which we can measure the importance of each objective. The first method discussed is the Pair-Wise comparison method developed by Saaty.

PAIR-WISE OBJECTIVE EVALUATION

Saaty (1972, 1977) has developed a procedure for obtaining a ratio scale for a group of elements based upon a paired comparison of each element. This method has also been used by Yager (1977) to obtain the values of subjective probabilities from a decision-maker.

Assume we have m objectives, and we want to construct a scale rating these objectives as to their relative importance to a decision-maker. We ask the decision-maker to compare the objectives in $m(m-1)/2$ paired comparisons. In particular, for each case where objective i is more important than objective j , a value a_{ij} is assigned from table 1.

Table 1. Level-of-importance definitions.

Level of Importance	Definition
1	Equal importance
3	Weak importance of one over the other
5	Strong importance of one over the other
7	Demonstrated importance of one over the other
9	Absolute importance of one over the other
2,4,6, 8	Intermediate values; between two adjacent judgments

Having obtained the above judgments, an m by m matrix B is constructed so that

1. $b_{ii} = 1$,
2. $b_{ij} = a_{ij}$, $i \neq j$,
3. $b_{ji} = 1/a_{ij}$.

Saaty then has shown that the eigenvector corresponding to the maximum eigenvalue of B is a cardinal ratio scale for the objectives compared.

As before, assume three information sources {Acoustic, Electronic, Infrared} are being rated on a scale as to their value in classifying ships:

Electronic is weakly more important than Acoustic. $\rightarrow a_{21} = 3$

Infrared is somewhere between equal and weakly more important than Acoustic. $\rightarrow a_{31} = 2$

Electronic is weakly more important than Intelligence. $\rightarrow a_{23} = 3$

Then our matrix B of compared comparisons is

$$B = \begin{matrix} & \begin{matrix} X & Y & Z \end{matrix} \\ \begin{matrix} X \\ Y \\ Z \end{matrix} & \begin{bmatrix} 1 & 1/3 & 1/2 \\ 3 & 1 & 3 \\ 2 & 1/3 & 1 \end{bmatrix} \end{matrix} .$$

We then solve the eigenvalue problem

$$BW = \lambda_{\max} W$$

and obtain the normalized eigenvector corresponding to λ_{\max} . For our example,

$$W = \begin{pmatrix} 0.16 \\ 0.59 \\ 0.25 \end{pmatrix}$$

where the components have been normalized to sum to 1. We now have a cardinal scale rating for the acoustic, electronic, and infrared information sources. In this case, our decision set becomes

$$D = A^{0.16} \Omega B^{0.59} \Omega C^{0.25}$$

or

$$\begin{aligned} D &= \{0.5, 0.7, 0.8\}^{0.16} \Omega \{0.8, 0.6, 0.7\}^{0.59} \Omega \{0.9, 0.5, 0.4\}^{0.25} \\ &= \{0.89, 0.95, 0.96\} \Omega \{0.88, 0.74, 0.81\} \Omega \{0.97, 0.84, 0.80\} \\ &= \{0.88, 0.74, 0.80\}. \end{aligned}$$

As before, the electronic emissions classification "overrides" the other two data sources and favors the decision of merchant ship as the most probable contact.

We now propose to use this same methodology to compare the importance of fuzzy constraints in the multi-objective decision problem. That is, we shall compare the objectives and obtain a scale rating for each of the objectives as to its importance. The difference is that instead of using the unit eigenvector, we shall use the eigenvector E ,

$$E = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_n \end{pmatrix} = nV = \begin{pmatrix} nw_1 \\ nw_2 \\ nw_n \end{pmatrix},$$

where V is the normalized eigenvector, because we want the average importance to be 1, so that if all the objectives are equally important, the $\alpha_i = 1$ for all i , and therefore all the $C_i^{\alpha_i}$ are not changed.

An Example

As an example, consider the following problem. Suppose a blue submarine is trailing a red submarine, and a maneuver is detected. The blue commander must react to this event by making a decision to *stop*, *turn*, or *no change*. In making the decision, the commander has the following objectives to satisfy:

1. Avoid Collision,
2. Avoid Detection,
3. Keep Contact,
4. Keep Station.

Suppose the commander evaluates the candidate decisions on a subjective basis with respect to the four objectives as follows:

$$C_1 = \{0.5, 0.7, 0.3\},$$

$$C_2 = \{0.5, 0.4, 0.8\},$$

$$C_3 = \{0.2, 0.01, 0.6\},$$

$$C_4 = \{0.6, 0.4, 0.9\}.$$

The numbers above can be considered the likelihood or probability that a given choice will satisfy a given objective. For example, the decision to *stop*, which is the first possible choice, C_1 , will result in a 50/50 chance of *avoiding a collision*. Hence, the first entry for C_1 is 0.5. Alternatively, based on a more subjective scale as suggested in table 2, we might say that there is a *moderate risk* of a collision if a submarine attempts to *stop* behind one that is maneuvering.

Table 2. A linguistic likelihood scale.

Term	Value
no change	0.0
unlikely	0.1
doubtful	0.2
uncertain	0.3
possible	0.4
moderate risk	0.5
significant chance	0.6
likely	0.7
probable	0.8
very probable	0.9
certain	1.0

Observe that the best decision is *no change* based on the min-max utility function,

$$\begin{aligned} D &= C_1 \cap C_2 \cap C_3 \cap C_4 \\ &= \{0.2, 0.01, 0.3\}. \end{aligned}$$

That is, the worst consequence under *no change* is a collision may not be avoided. However, the likelihood of this happening is not as great as losing contact, which is the worst consequence of the decision to *stop* or *turn*. Note that this procedure assumes that all of the objectives are equally important. This is usually not the case. In our example for instance, we would assume that avoiding a collision is normally much more important than the other objectives.

Accordingly, let us assume that each of the objectives is of differing degrees of importance. Each of the three objectives will be compared with each other as to importance in the selection of the candidate decision. A 4 by 4 matrix then is formed where element a_{ij} is the relative importance of objective C_i compared to objective C_j . Let us assume for our case the following Pair-Wise comparison of objectives reflects the commander's priorities (see table 1):

Avoid Collision has Strong Importance over Avoid Detection. $\rightarrow a_{12} = 5$

Avoid Collision has Weak Importance over Keep Contact. $\rightarrow a_{13} = 3$

Keep Contact has Demonstrated Importance over Avoid Detection. $\rightarrow a_{32} = 7$

Avoid Collision has Equal Importance to Keep Station. $\rightarrow a_{14} = 7$

Keep Station has Demonstrated Importance over Avoid Detection. $\rightarrow a_{42} = 7$

Keep Station has Weak Importance over Keep Contact. $\rightarrow a_{43} = 3$

A 4 by 4 B matrix of paired comparisons is formed where the element b_{ij} is the relative importance of objective C_i to C_j . Recalling that $b_{ji} = 1/a_{ij}$, we have

$$B = \begin{bmatrix} 1 & 5 & 3 & 1 \\ 1/5 & 1 & 1/7 & 1/7 \\ 1/3 & 7 & 1 & 1/3 \\ 1 & 7 & 3 & 1 \end{bmatrix}$$

The maximum eigenvalue for this matrix is 4.22, just slightly larger than $m = 4$, the number of objectives. The difference in these two values is a measure of inconsistency in the ratings. Saaty (1977) proposed using

$$s = [(\lambda_{\max} - m)/(2m - 2)]^{1/2}$$

as a test statistic for consistency. The nearer s to 0, the better; below 1 is probably OK; above 2 indicates severe inconsistency in the ratings. The consistency statistic in this example is 0.19, which is OK.

The eigenvector multiplied by the number of objectives becomes the set of weighting exponents. These are

$$\alpha_1 = 1.48$$

$$\alpha_2 = 0.20$$

$$\alpha_3 = 0.74$$

$$\alpha_4 = 1.58$$

indicating that *keeping station* is most important, followed closely by *avoiding collision*, whereas *keeping contact* and *avoiding detection* are not as important. This ordering is the relative importance to the person or persons (i.e., committee) making the paired comparisons. The relative importance can also vary according to the particular circumstances under which the decision is being made. For example, it may be more important to *avoid a collision* than *lose contact* in a cold war situation, but the reverse could be true in a hostile situation.

The decision model for this example becomes

$$\begin{aligned}
D &= C_1^{\alpha_1} \Omega C_2^{\alpha_2} \Omega C_3^{\alpha_3} \Omega C_4^{\alpha_4} \\
&= C_1^{1.48} \Omega C_2^{0.20} \Omega C_3^{0.74} \Omega C_4^{1.58} \\
&= \{0.5, 0.7, 0.3\}^{1.48} \Omega \{0.5, 0.4, 0.8\}^{0.20} \Omega \{0.2, 0.01, 0.6\}^{0.74} \\
&\quad \Omega \{0.6, 0.4, 0.9\}^{1.58} \\
&= \{0.36, 0.6, 0.17\} \Omega \{0.87, 0.83, 0.96\} \Omega \{0.3, 0.03, 0.68\} \\
&\quad \Omega \{0.45, 0.24, 0.85\} \\
&= \{0.3, 0.03, 0.17\}.
\end{aligned}$$

Thus *stop* is now the best decision. That is,

$$D = \max\{\min\{C_i^{\alpha_i}\}\} = \max\{0.3, 0.03, 0.17\} = 0.3.$$

Let us analyze this decision. In the previous example, we selected *no change*; here we select *stop*. The reason for this change in selection is that the importance of avoiding a collision now weighs heavily against our previous decision of *no change*. That is, the minimum rating of *no change* is reduced from 0.3 to 0.17 because of the importance of avoiding a collision. The lesser importance of keeping contact now makes *stopping* a better choice. That is, the minimum rating of *stop* is raised from 0.2 to 0.3 because keeping contact is reduced in importance. Notice also that the minimum rating of a *turn* also occurs for *keeping contact*, but this rating is so low, namely 0.01, that raising it to 0.03 made no difference.

Observations

Fuzzy sets provide a very fertile tool with which to investigate the multi-objective decision problem. One reason for this is the fact that by using a fuzzy set we are dealing in a very universal concept of *the degree to which an alternative satisfies an objective*, something that can be understood for any objective. A second reason is that fuzzy set theory provides a mathematical structure for manipulating vague ideas that become very common in complex multi-objective problems. Also, as in the case of decision-making under uncertainty (Miller & Starr, 1969), there are numerous valid ways in which to make decisions in multi-objective situations, the preference being a function of the decision-maker's nature.

The method discussed incorporates crucial concepts in multi-objective decision-making. First, the idea of comparing the objectives as to their importance incorporates to some degree an ability to account for trade-off effects between the objectives; more specific methods are necessary, and we hope will be developed. Second, the manner in which the power of each objective is included in the model corresponds to a hierarchical structure in the sense that each of the fuzzy sets corresponding to satisfaction of the objectives can be evaluated by various experts, and then these objectives can be compared by a next higher level in the decision process. Third, the methodology for evaluating the actual decision is basically a min-max procedure over multiple objectives.

RANK ORDER OBJECTIVE EVALUATION

The Zadeh and Bellman (1970) approach to multi-objective decision-making has the advantage of requiring only an ordinal evaluation or rank ordering of the preference information but the disadvantage of not allowing one to include the fact that the objectives differ in importance. The Yager (1977) method is just the opposite. It allows one to capture differing importances between objectives, but it requires stronger information on preferences. In this section, a model that has the advantages of both is suggested. This model will allow one to include the differing importance factor while still only requiring a rank ordering for preference information.

Assume that $\{S\}$ is the finite set of elements used to indicate preference information. Furthermore, assume that the only structure available on $\{S\}$ is a linear ordering. Let $Y = \{A_1, A_2, \dots, A_m\}$ be the set of objectives to be satisfied, and let $\{X\}$ be our set of alternatives. Assume that each objective is represented by a fuzzy subset of X with grades selected from S . Thus, for any $x \in X$, $A_i(x) \in S$ indicates the degree to which x satisfies the objectives specified by A_i . Let G be a fuzzy subset of Y in which $G(A_i) \in S$ indicates the importance of the objective A_i . We shall denote $G(A_i) = \alpha_i \in S$. Thus, we have for each objective a measure of how important it is to the decision maker for this decision.

Based upon the approach suggested by Yager, and other multiple objective methods that include importance, (Cochrane & Zeleny, 1973), (Keeney & Raiffa, 1977), (Starr & Zeleny, 1977), and (Zoints, 1978), a general form for this type of decision function is conjectured:

$$D(x) = M(A_1(x), \alpha_1) \text{ and } M(A_2(x), \alpha_2) \dots \text{ and } M(A_m(x), \alpha_m),$$

where $M(A_i(x), \alpha_i)$ indicates objective A_i evaluated at alternative x , modified by its importance. In Yager (1977) as previously discussed, it is suggested that $M(A_i(x), \alpha_i) = (A_i(x))^{\alpha_i}$. It has also been indicated that, however, this type of operation is not available to us when $A_i(x)$ and α_i are drawn from the finite linearly ordered set $\{S\}$. We must find some operation to replace this exponentiation. In a discussion of the implication operation (Yager, 1980) has shown that x^y is comparable to $x' \cup y$, which is the implication operation in two-valued logic. That is, they both generally act in the same manner. Whereas x^y requires more than an ordinal scale to implement, $x' \cup y$ needs only a finite linearly ordered set on which the appropriate negation can be defined. This leads us to conjecture that in S , $M(A_i(x), \alpha_i) = \alpha_i' \cup A_i(x)$. Thus, we are left with the conclusion that an appropriate model for including importances when our preferences are in S is

$$D = (\alpha_1' \cup A_1) \cap (\alpha_2' \cup A_2) \cap \dots \cap (\alpha_m' \cup A_m)$$

$$D = \bigcap_{i=1}^m (\alpha_i' \cup A_i) \quad ,$$

where $\alpha_i' \cup A_i = C_i$ is a fuzzy subset of X defined as follows: $C_i(x) = \alpha_i' \cup A_i(x)$. The optimal alternative is the $x \in X$ that maximizes D .

Let us examine this model to assure it is not being counterintuitive. First,

$$D = C_1 \Omega C_2 \Omega C_3 \Omega \dots \Omega C_m ,$$

where

$$D(x) = \text{Min} \{C_1(x), C_2(x), \dots, C_m(x)\} .$$

The implication of this is that the representative of x in D is selected as the $C_i(x)$ that has the smallest value. Thus, $\text{Min } C_i(x)$ becomes the most significant element in determining x 's contribution to the overall decision function D . We would hope that as an objective becomes more important it plays a more significant role in determining D . Recalling that $C_i(x) = \alpha_i' \cup A_i(x) = \text{Max} \{\alpha_i', A_i(x)\}$, consider the case when A_i is the least important objective; that is, $\alpha_i = 0$, the minimal element of S . Since negation is ordered revising, this implies $\alpha_i' = 1$ and hence $\text{Max} \{1, A_i(x)\} = 1 = C_i(x)$. Since $D(x) = \text{Min} \{C_i(x)\}$, it is very unlikely that $C_i(x)$ will be the determining value of $D(x)$. More generally, as the i th objective becomes more important, α_i increases, causing α_i' to get smaller, which in turn causes $\text{Max} \{\alpha_i', A_i(x)\} = C_i(x)$ to be decreasing and also increases the likelihood that $C_i(x) = A_i(x)$. Since $D(x) = \text{Min} \{C_i(x)\}$, it increases the possibility of $D(x)$ being determined by $A_i(x)$, the grade of membership of the most important objective. Furthermore, since the optimal alternative is x_{opt} such that

$$D(x_{\text{opt}}) = \text{Max } D(x) ,$$

$$x \in X$$

we see that for a given $y \in X$, if $A_i(y)$ is low in the more important objective, it is unlikely that y will be selected as the optimal solution. We can see that the proposed model satisfies our intuitive requirements and allows us to include importance measures defined by operations performed on linearly ordered finite sets.

An Example

Assume, as before, we must select a submarine maneuver from the set

$$X = \{\text{Stop, Turn, No Change}\} ,$$

given the four objectives

$$A = \{\text{Avoid Collision, Avoid Detection, Keep Contact, Keep Station}\}.$$

A set $S = \{s_i\}$ is designated to measure preferences. A particular example of a preference set is

$$S = \{\text{None, Very Low, Low, Medium, High, Very High, Perfect}\}$$

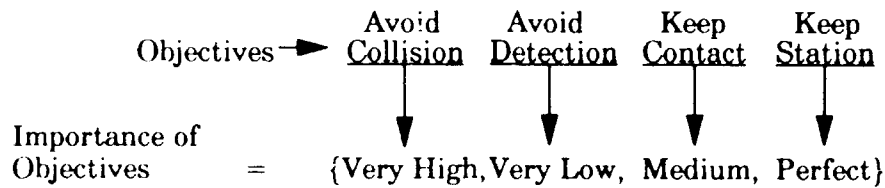
or, some numerical equivalent, such as

$$S = \{0, 1, 2, 3, 4, 5, 6\} .$$

First, we rate the alternatives with respect to the objectives:

	<u>Stop</u>	<u>Turn</u>	<u>No Change</u>
Avoid Collision = A_1	{Medium,	High,	Low}
Avoid Detection = A_2	{Medium,	Low,	Very High}
Keep Contact = A_3	{Very Low,	None,	High}
Keep Station = A_4	{High,	Low,	Perfect}

Next we evaluate the importance of each objective:



In our general notation,

$$\alpha_1 = \text{Very High}, \alpha_2 = \text{Very Low}, \alpha_3 = \text{Medium}, \alpha_4 = \text{Perfect},$$

or

$$\alpha_1 = s_5, \alpha_2 = s_1, \alpha_3 = s_3, \alpha_4 = s_6.$$

Since the negation in set S is order reversing, $S_i' = S_{6-i}$, then

$$\alpha_1' = s_1, \alpha_2' = s_5, \alpha_3' = s_3, \alpha_4' = s_0,$$

or

$$\alpha_1' = \text{Very Low}, \alpha_2' = \text{Very High}, \alpha_3' = \text{Medium}, \alpha_4' = \text{None}.$$

Since $C_i = a_i' \cup A_i$, we have

	<u>Stop</u>	<u>Turn</u>	<u>No Change</u>
$C_1 = \text{Very Low} \cup$	{Medium,	High,	Low}
=	{Medium,	High,	Low}
$C_2 = \text{Very High} \cup$	{Medium,	Low,	Very High}
=	{Very High,	Very High,	Very High}

			<u>Stop</u>	<u>Turn</u>	<u>No Change</u>
C ₃	=	Medium	U	{Very Low,	None, High}
	=			{Medium,	Medium, High}
C ₄	=	None	U	{High,	Low, Perfect}
	=			{High,	Low, Perfect}

Calculating $D(x) = \min_i \{C_i(x)\}$ we get

$$D = \begin{matrix} & \text{Stop} & \text{Turn} & \text{No Change} \\ \text{D} = & \{\text{Medium}, & \text{Low}, & \text{Low}\} \end{matrix}$$

Thus our selection based upon this model would be *stop*.

The basic reason for selecting *stop* is that its worst rating is for the objective of *keep contact*, but this objective is of only medium importance. *Stop* is selected over *turn* because *turn* gets a low rating in the most important objective, *keep station*. If the rating for *turn* in this objective were *medium* rather than *low*, it would then be tied with *stop*. Alternatively, if the objective *keep station* had only been of say, *medium* importance, then the effect of the low rating of *turn* in this objective would not be as significant, and the two would again tie. If, however, the objective of *avoid collision*, in which *no change* performed badly, were decreased in importance, this would then also have the effect of tying *no change* and *stop*.

A special procedure must be followed for selecting the optimal alternatives if we have ties. Assume that $x, y \in X$ are such that

$$D(x) = D(y) = \max_{z \in X} (D(z)) .$$

Since

$$D(x) = \min_i \{C_i(x)\} ,$$

there exists some k such that $C_k(x) = D(x)$. Similarly there exists some g such that $C_g(y) = D(y)$. Let $\hat{D}(x) = \min_{i \neq k} \{C_i(x)\}$, and let $\hat{D}(y) = \min_{i \neq g} \{C_i(y)\}$. Then, we

compare $\hat{D}(x)$ and $\hat{D}(y)$. If $\hat{D}(x) > \hat{D}(y)$, for example, we select x as our optimal. If, however, $\hat{D}(x) = \hat{D}(y)$, then there exists some r and e such that $\hat{D}(x) = C_r(x) = \hat{D}(y) = C_e(y)$. Then we formulate $\hat{\hat{D}}(x) = \min_{i \neq r} \{C_i(x)\}$ and $\hat{\hat{D}}(y) = \min_{i \neq e} \{C_i(y)\}$. We then compare $\hat{\hat{D}}(x)$ and $\hat{\hat{D}}(y)$. We continue in this

manner. If after exhausting all the objectives we still cannot distinguish between x and y , then they are deemed tied.

There exists an alternative approach to adjudicating ties. Since the scale used to measure our preferences was not a very fine scale, we may allow the decision maker to use a refinement of the scale to help in ties.

Assume x and y are tied for the optimal value in D . Thus, there exists some k such that $D(x) = C_k(x) = \text{Min } \{C_i(x)\}$, and there exists some g such that $D(x) = C_g(y) = \text{Min } \{C_i(y)\}$. We may now ask the decision maker to make a finer distinction between these two values. That is, though he/she selected the same $s \in S$ to evaluate $C_k(x)$ and $C_g(y)$, it may be possible for the decision maker to say that, for example, $C_k(x) > C_g(y)$. That is, it may be worthwhile to expend the effort to make a finer distinction between these two situations now that the decision has been reduced to selection based upon this information.

Observations

The model described above works in the following manner. For a particular objective, the negation of its importance acts as a barrier such that all ratings of alternatives that are below that barrier become equal to the value of that barrier. That is, we disregard all distinctions less than the barrier while keeping distinctions above this barrier. This works in the same manner as the classroom grading procedure of lumping all students whose grade averages fall below 60 into the F category while keeping distinctions of A, B, C, and D for students with grades of at least 60. In our model, however, this barrier varies, depending upon the importance of the objective. In particular, the more important the objective, the lower this barrier and thus, the more levels of distinction. Hence, as an objective becomes less important, we raise the distinction barrier, which penalizes the alternative less if it fails in this objective. In the extreme, if the objective is totally unimportant, then the barrier is raised to its highest value, and all alternatives are given the same rating, and no distinction is made based on this objective. If, however, the objective is most important, all distinctions are kept.

This model gives us an optimal solution. In particular, for any two alternatives x and y , if $A_i(x) \geq A_i(y)$ for all i , then $D(x) \geq D(y)$.

To see this, we note first that $D(x) = \text{Min } \{C_i(x)\}$, and $D(y) = \text{Min } \{C_i(y)\}$, and if $C_i(x) \geq C_i(y)$ for all i , then $D(x) \geq D(y)$.

Furthermore, since $C_i(x) = \alpha_i' \cup A_i(x)$, and $C_i(y) = \alpha_i' \cup A_i(y)$, then $A_i(x) \geq A_i(y)$ implies $C_i(x) \geq C_i(y)$. These two facts prove our observation.

Second, our solution is independent of irrelevant alternatives. That is, if we are given some set of alternatives X and find that our optimal solution is $x_{\text{opt}} \in X$ and then if we consider some extended set of alternatives $Y = X \cup Z$, our optimal solution will either be x_{opt} or some member of Z ; never some other member of X .

To see this, we note that if $D(x)$ is the value of x in our decision function when considering the set X , and if $E(x)$ is the value of x in our decision function when considering the alternate set Y , $D(x) = E(x)$ for all $x \in X$, and hence the $x \in X$ that is maximal over X is the same when considering X and Y . The only possible source of an alternative greater than x_{opt} is one in Z .

Finally, the model also has the property that the more important an objective is, the more significant its effect on D .

UTILITY FUNCTIONS

A utility function provides the mechanism to aggregate multiple objectives or constraints in the decision-making process. An open question is how to reflect the personal bias of the decision maker toward pessimism or optimism. Up to this point the min-max method has served as the utility function or aggregation operation in the fuzzy logic examples. This method selects as the best decision the one that minimizes the worst consequence of a decision. That is, the *and* or *min* operator selects the objective that is satisfied the least to judge a given alternative. This utility function implements a worst-case decision rule that is at the extreme of pessimism or conservatism. It might be considered most appropriate in a survival situation.

At the opposite extreme, the decision maker might require that a decision satisfy *at least one* of the objectives. This decision rule is implemented by "or"ing the objectives. This Pollyanna form of decision making assumes that the best result that can happen, will happen. Of course, we all know how often that happens. How many investors sold short before the recent market crash?

Most of us decision makers, like investors, fall somewhere between the two extremes mentioned above. It depends on the degree to which we are willing to gamble in a given situation. We might be satisfied if *most*, *many*, *at least half*, or *more than four* of the objectives are met. This suggests that selecting the appropriate utility function for a particular decision maker in a given situation is in itself a fuzzy decision problem. Several approaches to this problem have been taken. Zimmerman and Zysno (1980) have proposed an interesting but somewhat computationally intensive utility function called the *compensatory and*. This function allows for compensation for a low degree of membership in one constraint set by a higher degree of membership in another, whereas the logical fuzzy *and* operation corresponding to *min* does not.

Yager (1981) introduced yet another new utility function called ordered weighted average or weighted sum. An ordered weighted sum F is defined by

$$F(a_1, \dots, a_n) = \sum_i w_i b_i,$$

where

$$\sum w_i = 1$$

and b_i is the i^{th} largest element in the sorted collection $(a_1 \dots a_n)$. $B = (b_1 \dots b_n)$ is the vector of descending-order sorted elements of the arguments of F .

An Example

Consider the previous example where we had

	<u>Stop</u>	<u>Turn</u>	<u>No Change</u>
$C_1 =$	{Medium,	High,	Low}
$C_2 =$	{Very High,	Very High,	Very High}
$C_3 =$	{Medium,	Medium,	High}
$C_4 =$	{High,	Low,	Perfect}

or, in numerically equivalent terms,

$$C_1 = \{3, 4, 2\}$$

$$C_2 = \{5, 5, 5\}$$

$$C_3 = \{3, 3, 4\}$$

$$C_4 = \{4, 2, 6\}.$$

For the first alternative, *stop*, we have $F(a_1, a_2, a_3, a_4) = F(3, 5, 3, 4)$. Suppose the weight averaging or summing operator is $W = (0.1, 0.3, 0.2, 0.4)$. We form the ordered argument vector $B_1 = (5, 4, 3, 3)$ by sorting the a_i 's in descending order. Then applying W to obtain F we have

$$F(B_1) = (0.1*5 + 0.3*4 + 0.2*3 + 0.4*3) = 3.5.$$

We likewise compute F for each alternative to obtain

	<u>Stop</u>	<u>Turn</u>	<u>No Change</u>
$D =$	{3.5,	3.1,	3.7}

In this case, we would select *no change* as the most appropriate alternative.

Note $W = (1, 0, 0, 0)$ corresponds to pure "or"ing, while $W = (0, 0, 0, 1)$ corresponds to pure "and"ing. A pure weighted sum or mean operator would be $W = (1/4, 1/4, 1/4, 1/4)$. Thus, by proper choice of the W weight summing operator, we can vary the results of aggregation between the two extremes of *max* or "or"ing and *min* or "and"ing. While ordered weighted sum implements any degree of optimism or pessimism, an open question remains as how to obtain the weights that reflect the personal bias of the decision maker in a given situation. This is a topic of current research.

The complete algorithm described in appendices B and C gives the user a choice between "and"ing, which corresponds to *worst case* decision making, and pure weighted averaging, which corresponds to *expected case* decision making.

EXPLANATION FACILITY

The explanation facility makes the assumption that the user expects a particular alternative as the best choice. In particular, it is assumed that the weighted-sum method is a natural heuristic that approximates the user's expectations. When the user actually selects a weighted-sum method for a particular problem, the explanation is based on the two prominent criteria or objectives that influenced the decision. If another decision algorithm is selected, in particular, the min-max method, then calculations are done for the min-max method and the weighted-sum method. If the two algorithms agree, then the explanation is straightforward. If they disagree, then the explanation is based on the difference generated by the two algorithms. This technique is described further in the following section.

Explanation for the Min-Max Rank Method

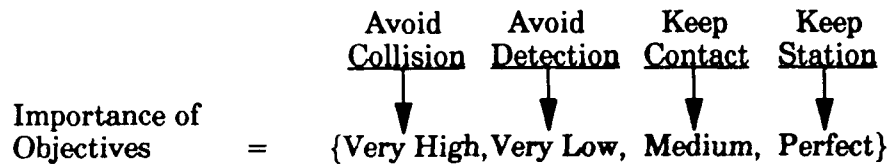
If a user's choice is also the choice selected by the min-max method, the explanation will just state that "Taking the importance of the objectives into account, alternative a_j has the highest overall rating; hence, it is the best choice." Explanation here is deliberately brief, since very little explanation is needed if the best choice is obvious.

If the user's expected choice is not selected by the method, the explanation facility will compare the expected choice with the choice selected by the method. For example, let a_j be the expected best alternative, and a_i be the actual best as selected by the method. A typical explanation would be "Alternative a_j is better than alternative a_i in objectives c_k, \dots, c_m but since alternative a_i satisfies objectives c_n, \dots, c_p more closely than alternative a_j , alternative a_i is the better choice."

For each rejected alternative, the explanation will output the reason for the alternative being rejected. For example, "Alternative a_j is rejected partially because of a low rating in objective c_k ." The explanation for rejected alternatives does not depend on whether the expected choice is actually the best choice or not.

As stated above, the explanation facility uses the weighted-sum heuristic to determine the expected best choice of the user. This is best explained by an example. Recall our previous example where we had the following input:

	<u>Stop</u>	<u>Turn</u>	<u>No Change</u>
Avoid Collision =	{Medium,	High,	Low}
Avoid Detection =	{Medium,	Low,	Very High}
Keep Contact =	{Very Low,	None,	High}
Keep Station =	{High,	Low,	Perfect}



In numerically equivalent terms we have

		<u>Stop</u>	<u>Turn</u>	<u>No Change</u>	<u>Importance of Objective</u>
Avoid Collision	=	{3,	4,	2}	5
Avoid Detection	=	{3,	2,	5}	1
Keep Contact	=	{1,	0,	4}	3
Keep Station	=	{4,	2,	6}	6

Taking the importance or rankings of the objectives into account, the min-max method transforms the input as follows: (see example in Rank Order Objective Evaluation section)

		<u>Stop</u>	<u>Turn</u>	<u>No Change</u>
Avoid Collision	=	{3,	4,	2}
Avoid Detection	=	{5,	5,	5}
Keep Contact	=	{3,	3,	4}
Keep Station	=	{4,	2,	6}

For this example, the decision set is

$$D = \{3, 2, 2\} ,$$

from which the algorithm would select *stop*. Now, for each alternative, we can compute its *weighted-sum* according to the following formula:

$$\text{weighted-sum} = x_1 + x_2 + x_3 + x_4 ,$$

where x_i = degree to which the alternative satisfies criterion i after the transformation. For this example, the weighted-sums for the alternatives are

Stop	=	3 + 5 + 3 + 4	=	15
Turn	=	4 + 5 + 3 + 2	=	14
No Change	=	2 + 5 + 4 + 6	=	17

No Change has the highest weighted-sum or the highest overall score; hence, the program will assume that *no change* is the expected choice of the user. The explanation generated for this case is as follows:

EXPLANATION

The No Change alternative is a better choice than Stop in satisfying the Avoid Detection, Keep Contact, and Keep Station objectives. But, since Stop satisfies the Avoid Collision objective more closely than No Change, Stop is the better choice.

The Turn alternative is rejected partially because of a low rating in the Keep Station objective. The No Change alternative is rejected partially because of a low rating in the Avoid Collision objective.

Explanation for the Min-Max Pair-Wise Method

The explanation facility will output the rating for each alternative and produce a natural-language comparison of the two alternatives with the highest ratings.

Consider the previous example in the section, Pair-Wise Objective Evaluation. The pair-wise algorithm generated the following ratings for the three alternatives:

Alternative	Rating
Stop	0.3
Turn	0.03
No Change	0.17

The top two choices are *stop* and *no change*. The explanation facility will just compare these two. The output will be

EXPLANATION

The No Change alternative is better than Stop in satisfying the Avoid Detection, Keep Contact, and Keep Station objectives. Stop is better than No Change in satisfying the Avoid Collision objective. With consideration to the overall importance of Avoid Collision versus Avoid Detection, Keep Contact, and Keep Station, the degree of superiority of Stop over No Change in Avoid Collision is deemed to be greater than the degree of superiority of No Change over Stop in Avoid Detection, Keep Contact, and Keep Station. Hence, Stop is preferred over No Change.

For cases in which the top two alternatives have the same or close-to-the-same ratings, the explanation will state such.

For example, suppose we had

Alternative	Rating
Stop	0.31
Turn	0.03
No Change	0.29

EXPLANATION

Since the difference between No Change and Stop is small, and due to the uncertainty of input, No Change and Stop are equally preferred.

The Pair-Wise method requires the user to make paired comparisons of all objectives. Often, inconsistencies arise from these comparisons. This is illustrated by the following example:

Let c_1 , c_2 , c_3 be the three objectives. The Pair-Wise comparisons are

1. c_1 is more important than c_2 with a degree 5.
2. c_2 is more important than c_3 with a degree 6.
3. c_3 is more important than c_1 with a degree 5.

Clearly, it is inconsistent that $c_1 > c_2$ and $c_2 > c_3$, but $c_3 > c_1$, where " $>$ " stands for *more important than*. These types of inconsistencies arise frequently, especially when the number of objectives becomes large. In fact, if there were m objectives, there can be up to $O((m,3))$ number of inconsistencies.

The explanation facility should warn the user when inconsistencies occur; however, it would be unwise to output all of the inconsistencies, since there can be such a large number of them. Instead, the explanation will try to find the two objectives that are *most responsible* for the inconsistencies. The facility will output only those inconsistencies in which the two objectives are directly involved.

The explanation facility will suggest that the relation between the two objectives be changed so the overall inconsistency can be reduced. For the case in which the changing of relation between the two objectives has no effect on the final ranking of alternatives, the explanation will not advise the user of the inconsistencies.

To continue with our previous example, suppose we have the following Pair-Wise comparisons:

Avoid Detection has Strong Importance over Avoid Collision. $\rightarrow a_{21} = 5$

Avoid Collision has Weak Importance over Keep Contact. $\rightarrow a_{13} = 3$

Avoid Collision has Equal Importance to Keep Station. $\rightarrow a_{14} = 1$

Keep Contact has Strong Importance over Avoid Detection. $\rightarrow a_{32} = 5$

Keep Station has Weak Importance over Avoid Detection. $\rightarrow a_{42} = 3$

Keep Contact has Equal Importance to Keep Station. $\rightarrow a_{34} = 1$

PAIRED-COMPARISON OF CRITERIA

	1	2	3	4
1	1	1/5	3	1
2	5	1	1/5	1/3
3	1/3	5	1	1
4	1	3	1	1

- 1 = Avoid Collision
- 2 = Avoid Detection
- 3 = Keep Contact
- 4 = Keep Station

There is inconsistency in the matrix of paired comparisons. The following list of inconsistencies is not exhaustive:

An inconsistency is as follows:

Keep Contact is more important than Avoid Detection by degree 5.

Avoid Collision is more important than Keep Contact by degree 3.

But Avoid Detection is more important than Avoid Collision by degree 5.

An inconsistency is as follows:

Keep Station is more important than Avoid Detection by degree 3.

Avoid Collision is more important than Keep Station by degree 1.

But Avoid Detection is more important than Avoid Collision by degree 5.

To minimize the inconsistencies listed above, it is recommended that the following change be made:

Avoid Collision is more important than Avoid Detection with degree

7 $a_{12} = 7$. Would you like to make this change? **yes**

```
*****
*   decision ratings range from 0 to 10   *
*           0  → very poor                *
*           10 → excellent                 *
*****
```

RANKED DECISION LIST

Alternative	Quantitative Decision-Value	Subjective Decision-Value
1. Stop	2.9	fair
2. No Change	1.8	poor
3. Turn	0.1	very poor

The mathematical basis for inconsistency analysis is developed in the section on Methods for Finding the Worst Inconsistencies.

Explanation for the Weighted-Sum Rank Method

The following is an example of the weighted-sum rank model. Suppose that the input is as before:

	<u>Stop</u>	<u>Turn</u>	<u>No Change</u>	<u>Importance of Objective</u>
Avoid Collision	= {3,	4,	2}	5
Avoid Detection	= {3,	2,	5}	1
Keep Contact	= {1,	0,	4}	3
Keep Station	= {4,	2,	6}	6

For each alternative, a weighted-sum rating is computed according to the formula:

$$\text{weighted-sum rating} = w_1 * c_1 + w_2 * c_2 + w_3 * c_3 + w_4 * c_4 ,$$

where c_i = degree to which the alternative satisfies objective i , and w_i = the ranking or weight of objective i .

For the above example, the weighted sums are

<u>Alternative</u>	<u>Rating</u>
Stop	$5*3 + 1*3 + 3*1 + 6*4 = 45$
Turn	$5*4 + 1*2 + 3*0 + 6*2 = 34$
No Change	$5*2 + 1*5 + 3*4 + 6*6 = 63$

In this case, *no change* has the highest weighted sum; it is selected by the weighted-sum method.

As before, the explanation facility makes the assumption that the user expects a particular alternative as the best choice. In particular, it further assumes that expected outcome of the user is based on the 2-weighted-sum heuristic. Again, this heuristic is explained by an example. Referring to our previous example, *keep station* and *avoid collision* are the two most important objectives. The 2-weighted-sum is computed for each alternative according to the formula:

$$2\text{-weighted-sum} = w_1 * c_1 + w_2 * c_2 ,$$

where the indices are taken over the two most important objectives.

That is, the 2-weighted sum picks the alternative satisfying the top two objectives best as the expected choice of user. (Note that if there are more than two candidates for the top two objectives, the algorithm will select the two that are nearest to the front of the objectives list; the rationale for such is that the user tends to read the objectives from left to right or top to bottom. The alternative that *does well early* will likely be selected by the user as the expected best choice.)

The result is

<u>Alternative</u>	<u>Rating</u>
Stop	$5*3 + 6*4 = 39$
Turn	$5*4 + 6*2 = 32$
No Change	$5*2 + 6*6 = 46$

No change has the highest 2-weighted-sum; hence, the algorithm will assume that it is the user's expected choice.

It just so happens that the algorithm picks *no change* as the best choice. The explanation is as follows:

EXPLANATION

Keep Station and Avoid Collision are 2 of the most important criteria, and since the No Change alternative satisfies Keep Station AND Avoid Collision best, No Change is the best choice.

Here, the explanation is deliberately brief, since the obvious best choice needs little explanation.

Suppose the weighted sum contradicts the 2-weighted-sum (i.e., the user's expectation). The explanation would be of the following form: (Assume that a_i is the actual-best choice, while a_j is the expected-best choice.)

a_j is better than a_i in c_k . But since a_i is better than a_j in c_l, c_m, c_n , a_i is the better choice.

In case of ties for top ratings, the explanation will state so, and list all the tied alternatives.

Explanation for the Weighted-Sum Pair-Wise Method

The explanation facility of this Method is the same as that of the Weighted-Sum Rank Method.

Observations

Simplicity and brevity are the chief objectives of the explanation facility. With this in mind, we have digressed from the usual tendency to output exhaustive traces of the procedural actions of the algorithm as explanation. Instead, we choose to make a few *reasonable* assumptions (such as, we assume that the user has *a priori* expectation of the outcome based on some heuristic, and we further assume that the heuristic used is the weighted-sum method discussed above) to prune the combinatorial number of comparisons among alternatives.

Furthermore, details of the inferencing have been omitted in the explanation. The reason is that the inference step is just a mathematical processing of the input, and it is not possible to describe mathematics exactly using natural language. With less than an exact description, the inference step would only confuse the user. Hence it is omitted. To compensate for the omission of inferencing, we recourse to the use of *white lies* to explain the outcome. *White lies* are approximations to explanations. They are not meant to give an exact account of the outcome. Rather, they are *more human readable* substitutes for the exact explanation.

The explanation facility has two shortcomings worth mentioning. First, the min-max algorithm picks one and only one alternative. In case of ties, even when they are identical, the algorithm will pick one arbitrarily. This action of the algorithm may not always be desirable. Methods for dealing with ties have been developed Yager (1981), but they have not been incorporated into the algorithm implemented here.

Secondly, the facility does not include some alternatives in the explanation, even though they are perfectly fine candidates. This drawback is a consequence of trying to satisfy our objective of brevity. Just imagine what it would be like if there are many alternatives, and each is mentioned in the explanation.

THE EIGENVECTOR METHOD OF FINDING WEIGHTS

This section discusses Saaty's method of obtaining a ratio scale based on paired comparisons of criteria. (See the discussion of the eigenvalue problem under the Pair-Wise Objective Evaluation section.)

The maximum eigenvalue and its corresponding eigenvector can be found to the desired accuracy by using an iterative computational process. However, we found the use of the approximation discussed in the next section to be considerably faster and sufficiently accurate for our purposes.

Approximation to the Maximum Eigenvalue and Normalized Eigenvector

The following approximation is valid when the user-provided matrix B of paired comparisons is reasonably consistent. The measure of inconsistency Saaty (1977, 239) for an m by m matrix B is

$$\text{inconsistency} = [(\lambda_{\max} - m)/(2m - 2)]^{1/2},$$

where λ_{\max} is the maximum eigenvalue of B. (A matrix is *consistent* only if $\lambda_{\max} = m$.) Saaty indicates that the consistency is acceptable when $(\lambda_{\max} - m)/(m - 1)$ is less than unity. Judging by examples shown later, we should keep this latter measure below about 0.2.

In slightly different notation, Saaty says that the solution "may be estimated by normalizing each column of B and taking the average over the resulting rows. This yields a vector W; in this case, one can readily obtain an estimate for λ_{\max} by computing BW, dividing each of the components of the resulting vector by the corresponding component of W, and averaging the results."

The vector W that results is the normalized eigenvector that Yager uses in his 1977 paper. The eigenvector is normalized by having its entries sum to unity. (Yager incorrectly calls this a unit vector.) Going through Saaty's description step-by-step, we have the following equations. The user's matrix of comparisons is

$$B = \begin{vmatrix} b_{11} & b_{12} & \dots & b_{1m} \\ b_{21} & b_{22} & \dots & b_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \dots & b_{mm} \end{vmatrix}$$

Normalizing each column of B gives \underline{B} , where

$$b_{ij} = b_{ij} / \sum_{i=1}^m b_{ij} .$$

Taking the average over the resulting rows yields W, the vector of weights.

$$W = \begin{vmatrix} w_1 \\ w_2 \\ \dots \\ \dots \\ w_m \end{vmatrix} , \text{ where } w_i = (1/m) \sum_{j=1}^m b_{ij} .$$

Computing BW and dividing each of the components of the resulting vector by the corresponding component of W gives

$$Y = \begin{vmatrix} y_1 \\ y_2 \\ \dots \\ \dots \\ y_m \end{vmatrix} , \text{ where } y_i = \sum_{j=1}^m b_{ij} \cdot w_j / w_i .$$

Averaging the results gives the estimate of the maximum eigenvalue.

$$\langle \lambda_{\max} \rangle = (1/m) \sum_{i=1}^m y_i$$

Example 1:

$$B = \begin{vmatrix} 1 & 1/3 & 1/2 \\ 3 & 1 & 3 \\ 2 & 1/3 & 1 \end{vmatrix} \rightarrow \underline{B} = \begin{vmatrix} 1/6 & 1/5 & 1/9 \\ 3/6 & 3/5 & 6/9 \\ 2/6 & 1/5 & 2/9 \end{vmatrix}$$

$$W = \begin{vmatrix} w_1 \\ w_2 \\ w_3 \end{vmatrix} \quad \begin{array}{l} w_1 = 43/270 = 0.15926 \\ w_2 = 159/270 = 0.58889 \\ w_3 = 68/270 = 0.25185 \end{array}$$

For convenience, let $w_i = w_i'/270$. Then $w_1' = 43$, $w_2' = 159$, $w_3' = 68$.

$$BW = \begin{vmatrix} w_1 + w_2/3 + w_3/2 \\ 3w_1 + w_2 + 3w_3 \\ 2w_1 + w_2/3 + w_3 \end{vmatrix}$$

$$Y = \begin{vmatrix} 1 + w_2'/3w_1' + w_3'/2w_1' \\ 3w_1'/w_2' + 1 + 3w_3'/w_2' \\ 2w_1'/w_3' + w_2'/3w_3' + 1 \end{vmatrix}$$

$$\begin{aligned} \lambda_{\max} &= [3 + 43(1/53 + 1/34) + (159/3)(1/43 + 1/68) + 68(1/86 + 3/53)]/3 \\ &= 3.0539 \end{aligned}$$

Estimated Values: $\langle \lambda_{\max} \rangle = 3.05$, $W = (0.159, 0.589, 0.252)$

True Values (Saaty, 1977, p. 266): $\lambda_{\max} = 3.05$, $W = (0.16, 0.59, 0.25)$

Example 2:

$$B = \begin{vmatrix} 1 & 9 & 7 \\ 1/9 & 1 & 1/5 \\ 1/7 & 5 & 1 \end{vmatrix} \rightarrow \underline{B} = \begin{vmatrix} 63/79 & 9/15 & 35/41 \\ 7/79 & 1/15 & 1/41 \\ 9/79 & 5/15 & 5/41 \end{vmatrix}$$

$$\begin{aligned} w_1 &= (63/79 + 9/15 + 35/41)/3 = 0.7504 \\ w_2 &= (7/79 + 1/15 + 1/41)/3 = 0.5989 \\ w_3 &= (9/79 + 5/15 + 5/41)/3 = 0.1897 \end{aligned}$$

$$BW = \begin{vmatrix} w_1 & 9w_2 & 7w_3 \\ w_1/9 & w_2 & w_3/5 \\ w_1/7 & 5w_2 & w_3 \end{vmatrix}$$

$$Y = \begin{vmatrix} 1 & 9w_2/w_1 & 7w_3/w_1 \\ w_1/9w_2 & 1 & w_3/5w_2 \\ w_1/7w_3 & 5w_2/w_3 & 1 \end{vmatrix}$$

$$\begin{aligned} \langle \lambda_{\max} \rangle &= [3 + w_1(1/9w_2 + 1/7w_3) + w_2(9/w_1 + 5/w_3) + w_3(7/w_1 + 1/5w_2)]/3 \\ &= 3.219 \end{aligned}$$

Estimated Values: $\langle \lambda_{\max} \rangle = 3.219$, $W = (0.750, 0.0599, 0.190)$

True Values: $\lambda_{\max} = 3.210$, $W = (0.77, 0.05, 0.17)$

Notes on the Eigenvector Method

Several pertinent excerpts concerning the eigenvector method are quoted or commented on here.

Number of Criteria

(Saaty, 1977, p. 234): "... the hierarchy serves as a useful tool for decomposing a large-scale problem, in order to make measurement possible despite the now-classical observation that the mind is limited to 7 ± 2 factors for simultaneous comparison." (Saaty discusses hierarchies in sections 4-6 of his paper and references Miller's paper entitled "The magical number seven plus or minus two: some limits on our capacity for processing information.") Saaty (1977, p. 251): "In general, informed judgment leads to better consistency. However, all the plots show that when the number of objects being compared exceeds 7 ± 2 , the consistency can be expected to be very poor — a theoretical confirmation of Miller's psychological observation. Later on we show how to overcome this limitation on the number of objects by using a method of hierarchical clustering."

Note that the number of paired comparisons for m criteria is $m(m-1)/2$.

m	2	3	4	5	6	7	8	9	10
No. pairs	1	3	6	10	15	21	28	36	45

Eigenvalue Existence

Saaty (1977, p. 235): "The Perron-Frobenius theory (Gantmacher, 1960) ensures the existence of a largest real positive eigenvalue for matrices with positive entries whose associated eigenvector is the vector of weights. This vector is normalized by having its entries sum to unity. It is unique."

Consistency Case

Adapted from Barbeau (1986, p. 14): If the matrix is consistent, then the columns are proportional to the normalized eigenvector (w_1, \dots, w_m) , and the rows are proportional to $(1/w_1, \dots, 1/w_m)$. Also, $b_{ij} \cdot b_{jk} = b_{ik}$ for all i, j, k . (For $m = 3$, this becomes $b_{XY} \cdot b_{YZ} = b_{XZ}$.)

METHODS FOR FINDING THE WORST INCONSISTENCIES

If the expert's matrix B shows substantial inconsistency, we would like to direct his/her attention to entries that most contribute to the high measure of inconsistency. The first method below detects the criterion whose ratings have the greatest overall inconsistency. The second method tries to pinpoint the one change in pair rating that will bring the matrix most closely to consistency. It suggests the replacement value of that pairing. If that change is unacceptable to the expert, he/she would

need to look at other pairs. If the inconsistency is evenly spread throughout the matrix, these methods will not help much.

The expert should initially be told that a rating of r means that a criterion is r times more important than another. Saaty (1977, p. 246) describes a rating of 3 as "Weak importance of one over another" and "Experience and judgment slightly favor one activity over another." Barbeau (1986, p. 17) explains 3 as "One moderately more important than the other" and "Experience and judgment slightly favor one option over the other." Consistency requires that the ratings be proportional to importance. Yager applies the weights with that interpretation in his proposed systems. We believe that the above descriptions of the rating 3 correlate poorly with "three times more important."

Column Method

Given: Inconsistent matrix B of pair-wise comparisons
Vector W of weights (normalized eigenvector of B)

1. Normalize the columns of B , giving matrix \underline{B} ,
2. For each column (j th) of \underline{B} , compute the squared distance

$$d(j) = (b_{1j} - w_1)^2 + (b_{2j} - w_2)^2 + \dots + (b_{mj} - w_m)^2$$

(If B is consistent, then $b_{ij} = w_i$ for all i and j .)

3. Reexamine the ratings for the criterion having the greatest value of $d(j)$. In particular, compare the two criteria having the two greatest values.

Note: The results will be exactly the same if rows (and inverse elements) are used instead of columns.

Example 1:

$$B = \begin{vmatrix} 1 & 4 & 1/4 & 1/3 \\ 1/4 & 1 & 1/2 & 3 \\ 4 & 2 & 1 & 3 \\ 3 & 1/3 & 1/3 & 1 \end{vmatrix} \quad \begin{array}{l} \lambda_{\max} = 5.38 \\ \text{inconsistency} = 0.48 \\ W = (0.21, 0.19, 0.41, 0.18) \end{array}$$

col. 1: $d(1) = 0.073$
col. 2: $d(2) = 0.152 \leftarrow$ worst
col. 3: $d(3) = 0.016$
col. 4: $d(4) = 0.077 \leftarrow$ next worst

The results suggest we should look at our ratings for pairings with the second criterion, especially with the fourth and first criteria.

Example 2:

$$B = \begin{vmatrix} 1 & 1/5 & 1/6 & 1/4 \\ 5 & 1 & 2 & 4 \\ 6 & 1/2 & 1 & 6 \\ 4 & 1/4 & 1/6 & 1 \end{vmatrix} \quad \begin{array}{l} \lambda_{\max} = 4.34 \\ \text{inconsistency} = 0.24 \\ W = (0.06, 0.45, 0.38, 0.12) \end{array}$$

col. 1: $d(1) = 0.036$
col. 2: $d(2) = 0.021$
col. 3: $d(3) = 0.034$
col. 4: $d(4) = 0.035$

We conclude that the inconsistency is distributed somewhat evenly throughout the ratings.

Example 3:

$$B = \begin{vmatrix} 1 & 5 & 3 & 1 \\ 1/5 & 1 & 1/7 & 1/7 \\ 1/3 & 7 & 1 & 1/3 \\ 1 & 7 & 3 & 1 \end{vmatrix} \quad \begin{array}{l} \lambda_{\max} = 4.225 \\ \text{inconsistency} = 0.194 \\ W = (0.37117, 0.04878, 0.18565, 0.39440) \end{array}$$

col. 1: $d(1) = 0.00439$
col. 2: $d(2) = 0.04367 \leftarrow \text{worst}$
col. 3: $d(3) = 0.034$
col. 4: $d(4) = 0.035$

Column 2 is much more inconsistent than the other three, which are about the same. Matings for criterion 2 should be reexamined.

Triple Method

Recall that a matrix is consistent if $b_{ij} \cdot b_{jk} = b_{ik}$ for all i, j, k . For an m -by- m matrix, the number of equalities to check can be reduced to $m!/3!(m-3)! = m(m-1)(m-2)/6$.

m	3	4	5	6	7	8	9
No. pairs	1	4	10	20	35	56	84

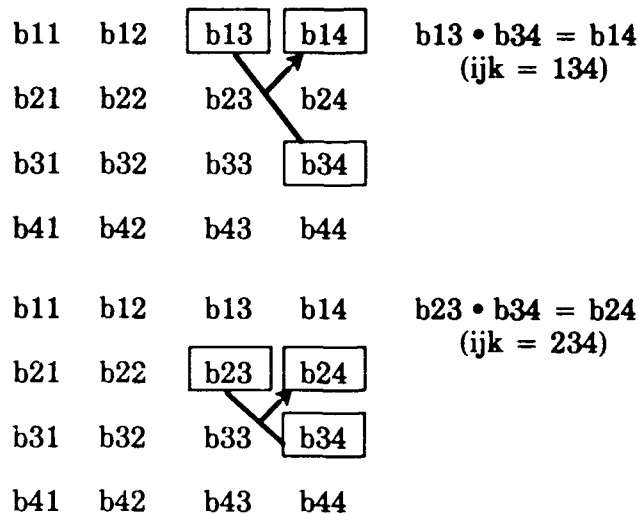
The ones for $m = 4$ are illustrated below.

b11	b12	b13	b14
b21	b22	b23	b24
b31	b32	b33	b34
b41	b42	b43	b44

$b12 \cdot b23 = b13$
($ijk = 123$)

b11	b12	b13	b14
b21	b22	b23	b24
b31	b32	b33	b34
b41	b42	b43	b44

$b12 \cdot b24 = b14$
($ijk = 124$)



One possible method of finding a single, large inconsistency is to take each of these *triples* and form the measure $m_{ijk} = b_{ij} \cdot b_{jk} / b_{ik}$. If the value is less than unity, let m_{ijk} be the inverse of this quantity. Examine the triple having the greatest value of m_{ijk} to see if equality can be reached or approached without badly affecting overlapping triples. If the two triples having the two greatest values of m_{ijk} overlap, this likely would pinpoint the worst inconsistency.

For examples, we use the same 4-by-4 matrices as in the column method.

Example 1:

$$B = \begin{bmatrix} 1 & \boxed{4} & \boxed{1/4} & \boxed{1/3} \\ 1/4 & 1 & \boxed{1/2} & \boxed{3} \\ 4 & 2 & 1 & 3 \\ 3 & 1/3 & 1/3 & 1 \end{bmatrix}$$

The resulting measures are

$$m_{123} = 8$$

$$m_{124} = 36$$

$$m_{134} = 2.25$$

$$m_{234} = 2$$

The worst case ($ijk = 124$) involves b_{12} , b_{24} , and b_{14} , while the next worst ($ijk = 123$) involves b_{12} , b_{23} , and b_{13} . Since these have b_{12} in common, we should look at the rating for the pair (criterion 1, criterion 2). In particular, we can algorithmically find the value that minimizes $m_{123} + m_{124}$ and recommend that change. First, the algorithm finds that $b_{12} = 1/2$ satisfies one equality, and $b_{12} = 1/9$ satisfies the

other. It computes $m_{123} + m_{124}$ for values of b_{12} between $1/9$ and $1/2$. Some partial results are

b_{12}	m_{123}	m_{124}	$m_{123} + m_{124}$
$1/6$	3.0	1.5	4.5
$1/5$	2.5	1.8	4.3
$1/4$	2.0	2.25	4.25
$1/3$	1.5	3.0	4.5

← Recommend $b_{12} = 1/4$

Note that the *worst* entry, b_{12} , is in the *worst* column identified by the first method.

Example 2:

$$B = \begin{bmatrix} 1 & 1/5 & 1/6 & \boxed{1/4} \\ 5 & 1 & 2 & \boxed{4} \\ 6 & 1/2 & 1 & \boxed{6} \\ 4 & 1/4 & 1/6 & 1 \end{bmatrix}$$

$$m_{123} = 2.4$$

$$m_{124} = 3.2$$

$$m_{134} = 4.0$$

$$m_{234} = 3.0$$

The measure m_{234} is almost as large as m_{124} and m_{134} , so we will look at the three triples having the greatest value. The two worst, triple-134 and triple-124, have b_{14} in common. (Columns 1 and 4 were the worst, although only slightly, based on the column method.) Triple-134 and triple-234 have b_{34} in common. Triple-124 and triple-234 have b_{24} in common. These three are all in the fourth column, which suggests that all ratings for criterion 4 should be reconsidered.

Example 3:

$$B = \begin{bmatrix} 1 & \boxed{5} & \boxed{3} & 1 \\ 1/5 & 1 & \boxed{1/7} & \boxed{1/7} \\ 1/3 & 7 & 1 & \boxed{1/3} \\ 1 & 7 & 3 & 1 \end{bmatrix}$$

$$m_{123} = 4.2$$

$$m_{124} = 1.4$$

$$m_{134} = 1$$

$$m_{234} = 3$$

Triple-123 and triple-234 have b_{23} in common. Triple-123 is satisfied by $b_{23} = 3/5$, and triple-234 by $b_{23} = 3/7$. The only rating value between $3/5$ and $3/7$ is $1/2$.

b_{23}	m_{123}	m_{234}	
<hr/>			
$1/2$	1.2	1.1667	← Recommend $b_{23} = 1/2$

This method identified b_{23} as being the most inconsistent element, while the column method identified column 2.

Based on these examples, the triple method seems preferable to the column method because it appears to give better results. Also, it does not require computing the weighting vector.

COMPARISON TO DEMPSTER'S RULE

Dempster's Rule

Dempster's rule of combination (Dempster, 1967) applies only to the combining of independent evidence. In this particular application, the m bodies of *evidence* correspond to the m criteria used to select an alternative, and the criteria must be reasonably independent. Dempster's rule is a generalization of Bayesian inference. Shafer (1976) later formulated Dempster's scheme within a flexible representation framework; thus, the popular label *Dempster-Shafer Theory*.

The use of Dempster's rule leads to upper and lower probabilities for each of n propositions. (In our application, these n propositions are our alternatives, a_1, a_2, \dots, a_n .) This set of propositions must be mutually exclusive and exhaustive. While Bayesian methods deal only with these n original propositions, Dempster's method can involve up to $2^n - 1$ *general propositions*. These are found by taking all possible disjunctions of the original n propositions. The most commonly used general proposition is the disjunction of all n propositions,

$$\phi = a_1 \vee a_2 \vee \dots \vee a_n,$$

where " \vee " denotes the Boolean OR. For $n = 3$, for example, the general propositions are

$$\begin{aligned}
b_1 &= a_1 = \sim (a_2 \vee a_3) \\
b_2 &= a_2 = \sim (a_1 \vee a_3) \\
b_3 &= a_3 = \sim (a_1 \vee a_2) \\
b_4 &= a_1 \vee a_2 = \sim a_3 \\
b_5 &= a_1 \vee a_3 = \sim a_2 \\
b_6 &= a_2 \vee a_3 = \sim a_1 \\
b_7 &= a_1 \vee a_2 \vee a_3 = \phi
\end{aligned}$$

where the tilde “ \sim ” denotes NOT.

An expert interprets each body of evidence, say the j^{th} , to provide a *probability mass assignment* M_j over the propositions. (Probability mass is simply probability. Shafer introduced the term *mass* because it is descriptively useful.) In the Bayesian case, there would be n probabilities summing to unity, and these would be equal in the face of total ignorance. In the generalized case, probabilities are assigned to any subset of the set of general propositions, also summing to unity. All of the probability mass is assigned to ϕ in the case of total ignorance.

Here we will show only the formula for the combining of two simple assignments. Formulas for a variety of cases are given in Dillard (1982a,b) and Dillard (1983a). An algorithm for computing the general case is in Dillard (1983a,b). In the simple formula below, the two probability mass assignments are Bayesian except for mass assigned to ϕ . This type of distribution arises in a number of applications, as described in Garvey (1981) and Dillard (1982a,b). Based on two bodies of evidence concerning the propositions, values are assigned to $M_1(a_1), M_1(a_2), \dots, M_1(a_n), M_1(\phi)$ and to $M_2(a_1), M_2(a_2), \dots, M_2(a_n), M_2(\phi)$. The combined distribution M is given by the following formulas.

$$\begin{aligned}
M(a_i) &= \{M_1(a_i) \cdot M_2(a_i) + M_1(a_i) \cdot M_2(\phi) + M_1(\phi) \cdot M_2(a_i)\} / C \\
&= F(a_i) / C,
\end{aligned}$$

where $F(a_i)$ represents the expression in braces and

$$C = M_1(\phi) \cdot M_2(\phi) + \sum_{i=1}^n F(a_i).$$

The resulting uncertainty is

$$M(\phi) = M_1(\phi) \cdot M_2(\phi) / C = 1 - \sum_{i=1}^n M(a_i).$$

The upper and lower probabilities of each a_i can be found for a combined distribution. The lower probability is called the *support* for the proposition, and the upper probability is called the *plausibility* of the proposition. In the case above, the support is $s(a_i) = M(a_i)$, and the plausibility is $p(a_i) = 1 - s(\sim a_i) = M(a_i) + M(\phi)$, where the support s of a proposition b is defined as follows:

$$s(b) = \sum_{b' \& b = b'} M(b') .$$

For example, $s(a_1 \vee a_4) = M(a_1) + M(a_4) + M(a_1 \vee a_4)$. A decision measure proposed in Dillard (1982a) is $s(a_i) - s(\sim a_i) = M(a_i) + p(a_i) - 1$, that ranges from -1 to 1.

For all applications of Dempster's rule, the combining can take place Pair-Wise or simultaneously and in any order; for example, $(M_1 \odot M_2) \odot M_3 = (M_2 \odot M_3) \odot M_1 = (M_1 \odot M_2 \odot M_3)$. If any assignment is Bayesian, the combined probability mass distribution will also be Bayesian. When all assignments are Bayesian, the combining operations reduce to standard Bayesian operations.


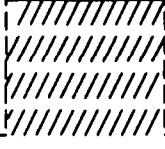
Examples of Dempster's Rule

Example 1. Suppose we have two alternatives, a_1 and a_2 , and two criteria (i.e., $n = 2$ and $m = 2$). Our two probability mass assignments are

Criterion 1: $M_1(a_1) = 0.1, M_1(a_2) = 0.4, M_1(\phi) = 0.5$

Criterion 2: $M_2(a_1) = 0.4, M_2(a_2) = 0.2, M_2(\phi) = 0.4$

The application of Dempster's rule is as shown below.

	0.1	0.4	0.5	
ϕ	a_1	a_2	ϕ	0.4
a_2		a_2	a_2	0.2
a_1	a_1		a_1	0.4
	a_1	a_2	ϕ	

The cross-hatched area is $0.1 \cdot 0.2 + 0.4 \cdot 0.4 = 0.18$, resulting in denominator $C = 0.82$. The resulting probability masses are then as follows:

$$M(a_1) = (0.1 \cdot 0.4 + 0.1 \cdot 0.4 + 0.5 \cdot 0.4) / 0.82 = 0.34146$$

$$M(a_2) = (0.4 \cdot 0.2 + 0.4 \cdot 0.4 + 0.5 \cdot 0.2) / 0.82 = 0.41463$$

$$M(\phi) = 0.5 \cdot 0.4 / 0.82 = 0.24390$$

Example 2. We have three alternatives, a_1 , a_2 , and a_3 , and two criteria. Our two probability mass assignments are

Criterion 1: $M_1(a_1) = 0.3$, $M_1(a_2) = 0.2$, $M_1(a_3) = 0.1$, $M_1(\phi) = 0.4$

Criterion 2: $M_2(\sim a_2) = 0.4$, $M_2(\phi) = 0.6$

		0.3	0.2	0.1	0.4	
ϕ	a_1		a_2	a_3	ϕ	0.6
$\sim a_2$	a_1			a_3	a_1	0.4
	a_1	a_2	a_3	ϕ		

The cross-hatched area is $0.2 \cdot 0.4 = 0.8$, resulting in denominator $C = 0.92$. The resulting probability masses are then as follows.

$$M(a_1) = 0.3 (0.4 + 0.6) / 0.92 = 0.32609$$

$$M(a_2) = 0.2 \cdot 0.6 / 0.92 = 0.13043$$

$$M(a_3) = 0.1 (0.4 + 0.6) / 0.92 = 0.10870$$

$$M(\sim a_2) = M(a_1 \vee a_3) = 0.4 \cdot 0.4 / 0.92 = 0.17391$$

$$M(\phi) = 0.4 \cdot 0.6 / 0.92 = 0.26087$$

Because a disjunction other than ϕ receives mass, we need to compute the support and plausibility (i.e., the lower and upper bounds) for the probabilities of the basic alternatives.

a_i	$(s(a_i), p(a_i) = 1 - s(\sim a_i))$	Decision Measure $s(a_i) - s(\sim a_i)$
a_1	(0.32609, 0.76087)	0.08696 ← Select a_1
a_2	(0.13043, 0.39130)	-0.47826
a_3	(0.10870, 0.54348)	-0.34783

Problem Domain Differences

We are using Dempster's rule in a way probably never envisioned by Dempster. Not only are we imposing a weighting procedure on the probability mass

assignments (but only when the assignments do not reflect criteria importance), we are applying his rule to a different kind of decision problem. Dempster intended his algorithm to be used to decide which of n mutually exclusive and exhaustive hypotheses is true. We are deciding which of n alternatives is best. Our alternatives are mutually exclusive mainly in that we will select just one. (In practice, the alternatives may be courses of action that have elements in common.)

Interpretation of the output probability mass distribution is also different for the two kinds of applications. For example, if plan a_5 is shown to be somewhat better than all other candidate plans, we choose a_5 without qualms. However, we have to be careful how we react to a decision that a contact is most likely a merchant ship and a little less likely to be a hostile cruiser.

Because of the difference between deciding which hypothesis is true and deciding which alternative to select, the assignment of probability to disjunctions has a somewhat different interpretation. These are examples of disjunctions when deciding which hypothesis is true:

Ship Classification Examples: (1) If the contact emits a signal peculiar to some radar system, we can assign probability to the disjunction of ship classes carrying that radar. (2) If a contact is maneuvering, we can assign probability to NOT-merchant, which is the disjunction of all other ship types. We let other kinds of evidence distribute that probability mass among all types but merchant.

Plan Selection Examples: (1) The Stop/Turn/No Change decision problem discussed earlier is a simple example of plan selection. (2) See the mission planning examples under the section "Applications of the Fuzzy Logic Decision Support Tool" below.

We should also note that many decision problems do not involve mutually exclusive hypotheses. For example, the Mycin system (Shortliffe, 1967) was designed to diagnose and select therapy for certain infectious diseases. The hypotheses are not exclusive, since the patient could have more than one infection. Shortliffe's confidence factors apply to decisions between two hypotheses (as opposed to the multi-alternative problem) so are well-suited to that type of application.

Dempster's Rule Applied to Weighted Assignments

Normally, the importance of the criteria should be considered when generating the probability mass assignments, and weighting is not needed. The weighting process is needed when the assignments are derived in the manner used in fuzzy decision aids. We are introducing the weighted application in order to compare performance with fuzzy decision aids.

Weighting every probability mass of an assignment M_j by w_j will accomplish nothing, since, when applying Dempster's rule, every product of probability masses would be weighted by the product $w_1 \cdot w_2 \cdot \dots \cdot w_m$, and the resulting distribution would be no different from the unweighted version. However, we can accomplish weighting by applying the weight w_j to every probability mass and then adding $1 - w_j$ to $w_j \cdot M(\phi)$. The adjusted distribution M_j for the j th criterion is then $M_j'(b) = w_j \cdot M_j(b)$ for $b \neq \phi$, and $M_j'(\phi) = w_j \cdot M_j(\phi) + 1 - w_j$.

We use Example 1 above with weights $w_1 = 0.5$ and $w_2 = 0.5$ to see if we get the same results as for the unweighted case. (Recall that the fuzzy set methods use weights normalized to sum to unity.) The adjusted assignments are

$$\text{Criterion 1: } M_1'(a_1) = 0.05, M_1'(a_2) = 0.2, M_1'(\phi) = 0.75$$

$$\text{Criterion 2: } M_2'(a_1) = 0.2, M_2'(a_2) = 0.1, M_2'(\phi) = 0.7$$

The combined probability mass distribution is

$$M'(a_1) = 0.20419, M'(a_2) = 0.24607, M'(\phi) = 0.54974.$$

Note that ϕ receives considerably more mass than for the unweighted case. The solution is to normalize the weights so that the maximum is unity.

$$w_j' = w_j/w_{\max}, \text{ where } w_{\max} = \max[w_j].$$

When the weights are all equal (i.e., $w_j = 1/m$ for all j), the process reduces to the usual use of Dempster's rule.

We use the same example of mass assignments for two weighted cases.

$$\text{Weighted Case 1: } w_1 = 0.2, w_2 = 0.8$$

$$\text{Weighted Case 2: } w_1 = 0.8, w_2 = 0.2$$

We expect that in case 1, alternative a_1 will receive considerably more probability mass than a_2 ; that is, we expect that $M(a_1) > M(a_2)$. In case 2, we expect that $M(a_2) > M(a_1)$. First we find the adjusted probability mass assignments.

Weighted Case 1 ($w_1' = 0.25, w_2' = 1$):

$$\text{Criterion 1: } M_1'(a_1) = 0.025, M_1'(a_2) = 0.1, M_1'(\phi) = 0.875$$

$$\text{Criterion 2: } M_2'(a_1) = 0.4, M_2'(a_2) = 0.2, M_2'(\phi) = 0.4$$

Weighted Case 2 ($w_1' = 1, w_2' = 0.25$):

$$\text{Criterion 1: } M_1'(a_1) = 0.1, M_1'(a_2) = 0.4, M_1'(\phi) = 0.5$$

$$\text{Criterion 2: } M_2'(a_1) = 0.1, M_2'(a_2) = 0.05, M_2'(\phi) = 0.85$$

Applying Dempster's rule to the adjusted mass assignments gives the following combined probability mass distribution:

$$\text{Weighted Case 1: } M'(a_1) = 0.38743, M'(a_2) = 0.24607, M'(\phi) = 0.36649$$

$$\text{Weighted Case 2: } M'(a_1) = 0.15183, M'(a_2) = 0.40314, M'(\phi) = 0.44503$$

As we expected, for case 1 we have $M(a_1) > M(a_2)$, and for case 2 we have $M(a_2) > M(a_1)$.

Before discussing the conversion of fuzzy measures into probability masses (for comparing the performance of Dempster's rule with fuzzy decision methods), we

summarize the adjusting of the probability mass distributions. Note that if mass is not given to ϕ or any other disjunction for the highest-weight case, the result will be a Bayesian probability distribution.

Weighting Assignments for Use in Dempster's Rule — A Summary

Given: • Alternatives a_1, a_2, \dots, a_n
 • m criteria and, for each (the j th)

—mass assignment M_j (independent of criterion importance)

—weight w_j (reflecting criterion importance)

Normalize the weights:

$$w_j' = w_j / w_{\max}, \text{ where } w_{\max} = \max_j \{w_j\}.$$

Adjust the probability mass assignments:

$$M_j'(b) = w_j' \cdot M_j(b), \text{ all } b \neq \phi$$

$$M_j'(\phi) = w_j' \cdot M_j(\phi) + 1 - w_j.$$

Apply Dempster's rule to the weighted assignments:

$$M' = M_1' \odot M_2' \odot M_3' \odot \dots \odot M_m'.$$

Conversion of Fuzzy Subsets into Mass Assignments

To convert the fuzzy set $C_j = (c_j(a_1), \dots, c_j(a_n))$ into a probability mass assignment M_j (for use in Dempster's rule), we first estimate $M_j(\phi)$. (For a Bayesian distribution, $M_j(\phi) = 0$.) How to best convert C_j into an equivalent probability mass assignment is a research issue. For now we do the following. We assign the remaining amount, $1 - M_j(\phi)$ to the alternatives in proportion to the corresponding c_j values, that is,

$$M_j(a_i) = (1 - M_j(\phi)) \cdot c_j(a_i) / \sum_i c_j(a_i).$$

Alternatively, we can exploit the flexibility of Dempster's rule by assigning probability to any disjunction of alternatives. For example, we may feel that one alternative a_2 is definitely poorer (judging for criterion j) than the other alternatives, but we do not wish to say the others are equally good compared to each other. In this case, we can assign $M_j(\sim a_2) = c$ instead of assigning $M(a_1) = M(a_3) = M(a_4) \dots = M(a_n) = c / (n-1)$ and $M(a_2) = d < c$.

In the communications system example that follows, we exercise this flexibility for one criterion. The resulting probability mass distribution is fairly equivalent to the C_j measures for a fuzzy algorithm.

Examples of Conversion

We must choose among four communications systems for a certain mission (i.e., alternative a_1 is sys_1 , a_2 is sys_2 , a_3 is sys_3 , and a_4 is sys_4). Our three criteria are communications range, ease of use, and reliability. We have previously computed the weighting vector $W = (w_1, w_2, w_3, w_4)$.

Criterion 1: Communications Range

Assume that all meet the minimum requirements. The fuzzy membership value is chosen to be the percentage of time that high-quality communications are maintained without having to move closer. If not already known, these percentages can be computed based on a number of known parameters and distribution functions. Fuzzy set C_1 indicates how well each of the systems satisfies this range criterion.

$$C_1 = (c_1(sys_1), \dots, c_1(sys_4)) = (0.55, 0.60, 0.50, 0.85) .$$

In specifying a criterion's probability mass assignment for use in Dempster's rule, we can express our uncertainty about our accuracy by assigning some of the probability to the disjunction ϕ of all alternatives. In this example, $\phi = sys_1 \vee sys_2 \vee sys_3 \vee sys_4$. Since we have very good information about range performance, our uncertainty here is about $M(\phi) = 0.1$. We convert the measures in C_1 to a probability mass distribution by proportionally dividing the remaining 0.9 probability among the four alternatives, that is, let $M_1(a_i) = 0.9(c_1(a_i) / \sum c_1(a_i))$. We then have $M(\phi) = 0.1$, $M_1(sys_1) = 0.198$, $M_1(sys_2) = 0.216$, $M_1(sys_3) = 0.18$, and $M_1(sys_4) = 0.306$.

In practice, the expert is unlikely to assign such precise numbers, but we wish to illustrate how to make them equivalent to the fuzzy measures for the sake of comparison.

Criterion 2. Ease of Use

This is a subjective judgment. Assume that sys_1 and sys_4 are older and somewhat clumsy compared to sys_2 and sys_3 . Fuzzy set C_2 indicates how well each communications set meets the ease-of-use criterion.

$$C_2 = (c_2(sys_1), \dots, c_2(sys_4)) = (0.40, 0.90, 0.90, 0.40) .$$

In specifying the probability mass distribution for criterion 2, we first set the value of $M_2(\phi)$. Suppose we feel that our uncertainty on the matter rates a value of about 0.3. If we use the same conversion method as before, we have the following distribution.

$$M_2(\phi) = 0.3, M_2(sys_1) = 0.1077, M_2(sys_2) = 0.242,$$

$$M_2(sys_3) = 0.242, M_2(sys_4) = 0.1077 .$$

Criterion 3: Reliability

Assume that the older communication systems, sys_1 and sys_4 , have proven to be highly reliable in the past, but, because of their age, we cannot expect this to continue. Our brief experience with sys_2 indicates that it is probably highly reliable.

Although we cannot be sure of this, our doubt is offset by the fact that the components will not fail from age. Several users have reported failure of one of the components of their new sys3's, so we are somewhat concerned about its reliability.

Arbitrarily, we decide a 10-percent failure rate is intolerable, and we let

$$c_2(\text{sys}_i) = 1 - 10 \cdot \text{prob}(\text{system } i \text{ fails during mission}).$$

Fuzzy set C_3 regarding reliability is

$$C_2 = (c_2(\text{sys}_1), \dots, c_2(\text{sys}_4)) = (0.95, 0.95, 0.60, 0.95).$$

We convert these measures into probability masses, specifying $M_3(\phi) = 0.4$. We then have $M_3(\text{sys}_1) = M_3(\text{sys}_2) = M_3(\text{sys}_4) = 0.1652$, and $M_3(\text{sys}_3) = 0.10435$.

At this point, the user of Dempster's rule, based on his knowledge (or lack of knowledge) of reliability figures, may feel more comfortable making the following assignment.

$$M_3(\phi) = 0.6, M_3(\sim \text{sys}_3) = 0.4$$

This is fairly consistent with the distribution obtained by conversion. Note that $\sim \text{sys}_3 = \text{sys}_1 \vee \text{sys}_2 \vee \text{sys}_4$, and that $M_3(\text{sys}_1) + M_3(\text{sys}_2) + M_3(\text{sys}_4) - M_3(\text{sys}_2) = 0.391$.

Experimental Comparisons

The Yager (1977) weighted min-max method was compared with the weighted version of Dempster's rule for several examples. The fuzzy subsets used in the min-max method were converted into mass assignments in the manner described above. The results were as expected for the cases considered. The two methods generally resulted in the same decision. When the two methods gave different results, the reason was that the decision alternative having the maximum value of its minimum measure had low values of its other measures, relative to the measures for the other alternatives.

We cannot conclude that one method is better than the other, especially with so few data points. The choice of which method is better is a matter of philosophical preference and of practicality. The method based on Dempster's rule has the advantage that all input data are fully exploited. It is also more flexible in that probability assignments can be given to disjunctions of alternatives. The fuzzy method has a considerable advantage in simplicity.

APPLICATIONS OF THE FUZZY LOGIC DECISION SUPPORT TOOL

During the past several years, fuzzy logic has found numerous applications in fields ranging from finance to earthquake engineering. The most important and striking application has been the realm of fuzzy-logic-based process control. Among the well-publicized applications are automatic train operation, vehicle control, robot

control, speech recognition, universal controllers, and stabilization control. Maier and Sherif (1985) have published a survey of applications of fuzzy logic.

Of interest here is the application of fuzzy logic to command and control. In general the problem can be posed as the generation, evaluation, and/or selection of the good plans or courses of action given a situation and the rules of engagement. As in our running example, given the situation of a submarine trailing activity, three courses of action were considered (stop, turn, no change) for a given set of objectives (avoid collision, avoid detection, keep contact, and keep station).

Prototype applications of the fuzzy logic decision tool to problems in mission planning for air strikes are described here. For further information, see Larsen (1989).

A PLAN SELECTION AID FOR THEATER STRIKE MISSION PLANNING

The overriding joint-service need in C3 is rapid, reliable, and effective exchange of timely information for planning, decisions, and command action. Project Juniper addresses the technology of distributed expert decision aids in the context of cooperative and supportive joint Navy/Air Force air strike mission planning. Air strike mission planning is currently a time-consuming and worker-intensive process. To respond to rapidly changing battle environments, the time to accomplish all levels of mission planning must be significantly reduced. An increasingly lethal threat environment requires that planning effectiveness must also be improved.

The Navy and Air Force have developed expert systems, the **Air Strike Planning Advisor (ASPA)** and the **Knowledge-based Replanning System (KRS)**, respectively, that support the planning of strikes against land targets. These developments use different hardware and software, but they have been networked to demonstrate the feasibility of distributed planning for joint Navy/Air Force air strike missions. That is, the system users work together, sharing data and results, to plan a joint mission.

KRS is a prototype decision aid aimed at exploring the issues involved in automated and semi-automated mission planning. KRS deals primarily with the allocation and scheduling of resources for strike missions and the associated support missions such as air escort, surface-to-air missile suppression, electronic countermeasures, and refueling. In addition, KRS includes Tomahawk Land Attack Missile (TLAM) missions. KRS can be used interactively by the user for database query, plan verification, or plan generation at the theater level. A Navy version and an Air Force version of KRS work together to plan joint missions in various theaters of operation. When standard plan knowledge cannot be used at the theater level, the Navy KRS can task ASPA to automatically develop subplans tailored to new targets, a changing threat environment, or changing weather conditions.

The top-level interface of the Juniper system allows the user to generate and send joint tasking directives or target lists to the respective Navy and Air Force planning systems. When strike plans are developed by the individual services, they are then integrated at the top level by a process of machine-assisted review and comparison using a prototype version of the fuzzy logic decision tool, referred to as the Plan

Selection Aid (PSA). The purpose of PSA is to provide a decision aid to help a user choose between proposed candidate missions. The techniques of multiple-alternative, multiple-criteria decision making, based on fuzzy set theory, are employed to suggest the most desirable candidate mission, given previously defined and weighted criteria. The user may select his/her own plan evaluation criteria and their relative importance. The joint plan is determined by the best combination of Navy/Air Force assets needed to accomplish the primary and support missions, considering primary mission parameters such as (1) availability of assets like aircraft, weapons, and fuel; (2) number of assets required, including refueling and other support assets; (3) mission duration and time over target; and (4) likelihood of mission success. A succinct explanation of the reasoning behind this suggestion is provided to the user.

The PSA system used the weighted-sum utility function with the rank order objective evaluation method. PSA is directly integrated into the Juniper system on a Symbolics 3600 in ZetaLISP. The PSA display uses windows, menus, multiple fonts, and icons as well as the natural language explanations. The decision aid is broken into three steps to allow the operator maximum flexibility and feedback. Plan information is carried from step to step and is always available for operator reference.

When the operator reaches the point in the Juniper system where candidate mission plans have been developed, he/she may choose to enter PSA. Figure 1 shows an example summarizing various candidate missions that have been planned against the petroleum, oil, and lubrication (POL) facilities at Tripoli.

Upon entering PSA, the user selects which plans or plan components to compare and which template, if any, to use. In this example, the user has chosen to compare strike plan components for a Navy and Air Force mission and has chosen the preemptive strike template (see figure 2). A template is a predefined set of weighted criteria that may be used or altered by the user. For a preemptive strike, survivability has a very high rating while sustainability has a very low rating.

The weighted selection criteria are then displayed with the operator given the opportunity to modify the weights (see figure 3). The candidate plans are then given a score with respect to each criteria. This internal evaluation is done by a rule and heuristic-based approach. The following is an example of a rule:

```
If night or poor visibility,  
then  
    increase survivability of F-111 by 2,  
    increase survivability of A-6 by 1,  
    decrease survivability of other TACAIR  
        by 3 if low threat or  
        by 6 if high threat.
```

Once the weighted criteria and the plan evaluations are obtained, PSA performs the mathematics that culminates in a proposed candidate plan. The recommended plan is then presented to the operator along with a brief explanation (see figure 4). The operator is asked to choose between the candidate plans. Once a plan has been selected, it is added to the Joint Plan Summary in Juniper, and the losing candidate's resources are deallocated. The operator can return to PSA's top level to examine another plan component, or the operator can return to Juniper.

JUNIPER									
HELP		PSA		RESTART		DISPLAY OPS		VIEWPORT	
TARGET NAME	PK	TOT	PRIORITY	COMMENT					
SIDI-BILAL	0.9	1700	4	Naval & Comando Base					
SERVICE	MAC	AC	MAC-AVAIL	BASE	PK	REFUEL	RT-MILES	STATUS	
Air Strike Mission	10	A-6E	0	AMERICA	0.9	0	219	SUCCESS	
NAVY									
Air Strike Mission	2	F-4E	23	ALCONBURY	0.91	2	1307	SUCCESS	
AIR FORCE									
<p>JUNIPER ALTERNATIVE PLANS</p> <p>Displaying strike plans from NAVY KRS & AF KRS for comparison.</p> <p>Each plan is mouse-sensitive.</p> <p>Mouse LEFT on a plan to SELECT it.</p> <p>Mouse MIDDLE on a plan to DELETE it which will in turn deallocate the resources used for it.</p> <p>All plans must be selected or deleted before another target is planned.</p>									
STATUS & EXPLANATION WINDOW									

Figure 1. Candidate missions.

PLAN SELECTION AID					
STEP ONE		STEP TWO		STEP THREE	VIEWPORT
HELP					
TARGET NAME	PK	TOT	PRIORITY	CURRENT PLAN	COMMENT
6101-91LAL	8.9	1700	4	Naval & Conando Base	
PLAN COMPONENT			TEMPLATE SELECTION		
<p>Strike</p> <p>SSM</p> <p>ECM</p> <p>AEM</p> <p>Refuel</p> <p>All</p> <p>A Strike plan has been generated by both the Air Force and the Navy</p>			<p>Preemptive</p> <p>Sustained</p> <p>Rollback</p> <p>Covert</p> <p>Blank</p> <p>x</p> <p>A preemptive strike implies maximal survivability. Sustainability is not important.</p>		
<p>STATUS & EXPLANATION WINDOW</p> <p>12/06/00 16:55:24 JUNIPER</p> <p>USER: 12/ 3600-LOGIN serving JUPITER</p>					

Figure 2. PSA Step One.

PLAN SELECTION AID						
STEP ONE		STEP TWO		STEP THREE		VIEWPORT
HELP	CURRENT PLAN		PLAN COMPONENT		TEMPLATE	
TARGET NAME	PK	TOT	PRIORITY	COMMENT	STRIKE	PREEMPTIVE
6101-BILAL	0.9	1700	4	Naval & Conando Base		
SELECTION CRITERIA			PLAN EVALUATION			
CRITERIA	IMPORTANCE			NAVY	AIR FORCE	
Cost effective	Medium			Medium	Very High	
Sustainable	Very Low			Low	Very Low	
Survivable	Utmost			High	LOW x	
Surprise	High			Medium	Very Low	
			With night missions or low-visibility and a SAM threat, the mission survivability is significantly reduced if using TACAIR other than F-111s or A-6s.			
<div> <div>STATUS & EXPLANATION WINDOW</div> <div>12/06/80 17:00:48 JUNIPER</div> </div> <div> <div>USER:</div> <div>3600-LOGIN serving JUPITER</div> </div>						

Figure 3. PSA Step Two.

PLAN SELECTION AID						
HELP		STEP ONE		STEP TWO		VIEWPORT
		CURRENT PLAN				
TARGET NAME	PK	TOT	PRIORITY	COMMENT	PLAN COMPONENT	TEMPLATE
SIDI-BILAL	0.9	1700	4	Naval & Conando Base	STRIKE	PREEMPTIVE
RECOMMENDED PLAN				SELECT PLAN		
<p>The decision based on this model is NAVY.</p> <p>In this model the criteria SURVIVABLE and SURPRISE are judged to be the most important. Since the alternative NAVY satisfies the criteria SURVIVABLE and SURPRISE better than the other alternatives, NAVY is considered the best choice.</p>				<p>Which STRIKE plan do you want to select?</p> <p><input checked="" type="checkbox"/> Navy X</p> <p><input type="checkbox"/> Air Force</p>		
<div>STATUS & EXPLANATION WINDOW</div> <div>12/06/88 17106146 JUNIPER</div> <div>USER: 141</div> <div>3698-LOGIN serving JUPITER</div>						

Figure 4. PSA Step Three.

PLAN GENERATOR AND EVALUATOR FOR THE AIR STRIKE PLANNING ADVISOR

The Air Strike Planning Advisor (ASPA) functionality currently includes weaponizing and defense suppression planning. The weaponizing module integrates existing stand-alone decision aids, large data files and expert knowledge to support the selection of aircraft, weapons loads, load configuration, and delivery tactics to achieve desired strike mission effectiveness. The defense suppression module is designed to support a strike leader in the selection of resources and tactics to suppress the local area defensive threat in support of a land air strike. The defensive threat includes early warning radars, surface-to-air missiles, anti-aircraft artillery, and air intercept. The resources and tactics to be considered include the electronic countermeasures, decoys, antiradiation missiles, and deception employed by Naval aircraft.

A prototype fuzzy logic decision tool has been incorporated into the defense suppression planner to both generate and evaluate plans. Existing computer aids that evaluate the effectiveness of a defense suppression plan would employ a Monte Carlo model that simulates a given scenario and produces a probability of survival for the aircraft. Such calculations are obviously unappealing to the strike group and may not even be meaningful because of all the assumptions involved and the vagaries of random number generators.

The prototype fuzzy logic decision tool in ASPA gives the strike leader a means of subjectively assessing a plan or a set of alternative plans according to the leader's criteria. These criteria may involve measures of performance such as those listed in table 3. In addition, a plan may be evaluated using criteria such as the expected number of valid missile firings rather than the probability of survival. It is then up to the strike group to determine the risk involved based on their experience.

Figure 5 is a conceptual design for a deception plan generator. The entries in a multidimensional matrix are rule sets that generate a score for each deception hardware capability, each tactic, and each evaluation criterion. There is a set of rule sets for each possible situation. This is depicted by the multiple levels or planes of rule sets in figure 5.

After the user inputs a description of a given situation, the appropriate rule set is used to score each hardware/tactic option for each selection criterion. The scores are then weighted according to the importance of the selection criteria, and a plan is recommended.

Table 3. ASPA defense suppression: plan evaluation criteria.

Fuzzy Set Logic Methodology

- Subjective ranking of criteria
- Objective measurements of criteria

CRITERIA

MEASUREMENT

EA-6B	Optimal positioning with respect to EW & GCI
Missile systems	Probability of kill (Pk) or safety factor
Chaff/flares	% of area covered
DECM	# of missile systems in threat environment
Fuel	Amount used
ACFT	% of available aircraft used
ARMs	% of missiles on CV used

Output: Ordered ranking of plans
Summary explanation

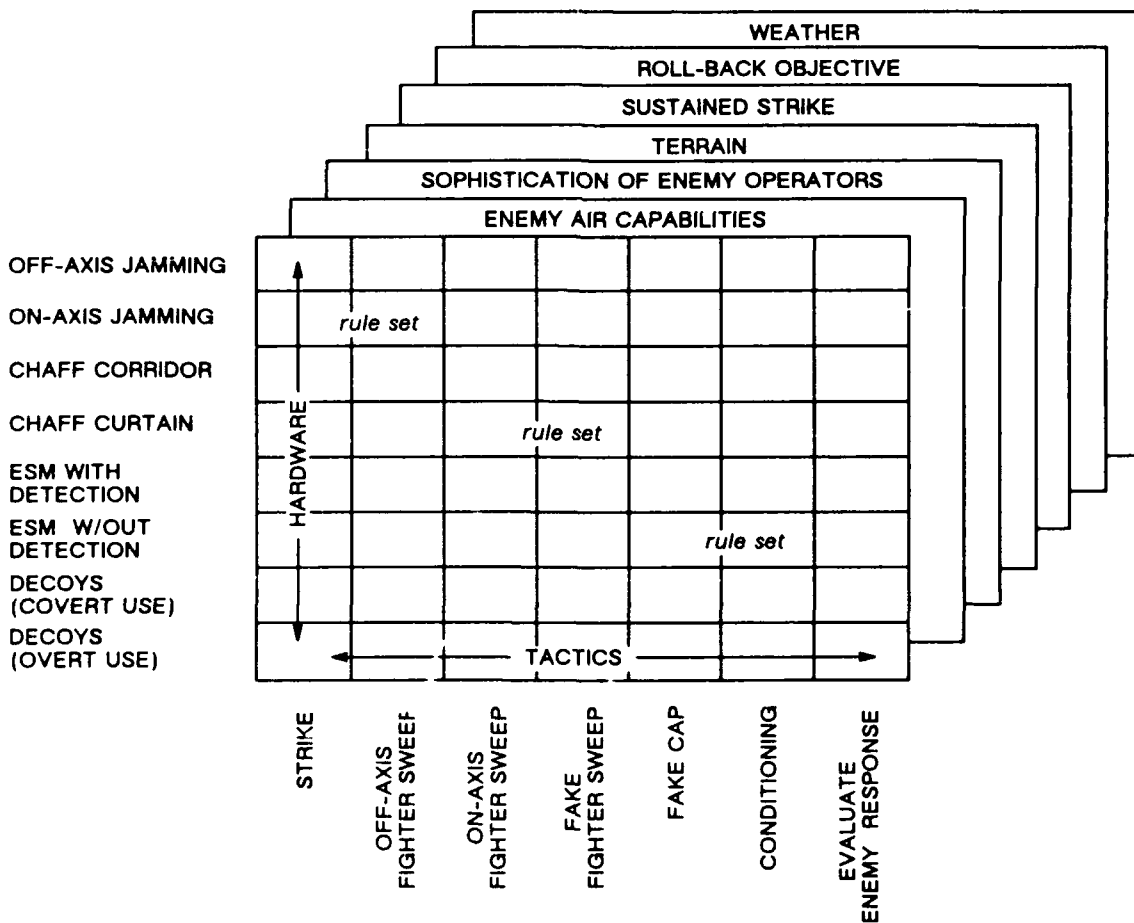


Figure 5. Deception plan generator.

REFERENCES

- Barbeau, E. 1986. "Perron's Results and a Decision on Admissions Tests," *Mathematics Magazine*, vol. 59, no. 1 (February), pp. 12-22.
- Cochrane, J. L. and M. Zeleny. 1973. *Multiple Criteria Decision Making*. Univ. of South Carolina Press, Columbia, SC.
- Dempster, A.P. 1967. "Upper and Lower Probabilities Induced by a Multivalued Mapping," *Annals of Mathematical Statistics*, no. 38, pp. 325-339.
- Dillard, R.A. 1982a. *Computing Probability Masses in Rule-Based Systems*, NOSC TD 545. Naval Ocean Systems Center, San Diego, CA.
- Dillard, R.A. 1982b. "The Dempster-Shafer Theory Applied to Tactical Data Fusion in an Inference System," *Proc. of the Fifth MIT/ONR Workshop on C3 Systems*, Monterey, CA, Aug. 23-27, pp. 170-174.
- Dillard, R.A. 1983a. *Computing Confidences in Tactical Rule-Based Systems Using Dempster-Shafer Theory*, NOSC TD 649. Naval Ocean Systems Center, San Diego, CA.
- Dillard, R.A. 1983b. "Tactical Inferencing with the Dempster-Shafer Theory of Evidence," *Proc. of the 17th Asilomar Conf. on Circuits, Systems and Computers*, Pacific Grove, CA, Oct. 31-Nov. 2, pp. 312-316.
- Dockery, J. 1987. "C3 Managerial Decision Aids for Dealing with Uncertainty-Variou Approaches," *Conf. Proc., Workshop on Assessing Uncertainty*, (held at the Naval Postgraduate School, Monterey, CA, Nov. 1986). Dept. of Statistics, Stanford Univ., Stanford, CA.
- Garvey, T.D., J.D. Lowrance, and M.A. Fishler. 1981. "An Inference Technique for Integrating Knowledge from Disparate Sources," *Proc. IJCAI 7*, vol. 1, pp. 319-325.
- Genest, C. and J.V. Zidek. 1986. "Combining Probability Distributions: A Critique and Annotated Bibliograph," *Statist. Science*, vol.1, no. 1, pp. 114-134.
- Heshmaty, B. and A. Kandel. 1985. "Fuzzy Linear Regression and its Applications to Forecasting in Uncertain Environment," *Fuzzy Sets and Systems*, vol. 15, no. 2 (March), pp. 159-190.
- Keeny, R. L. and H. Raiffa. 1976. *Decisions with Multiple Objectives*, Wiley, New York.
- Larsen, R. W. 1989. *Project Juniper: Cooperative Joint Mission Planning*, NOSC TR 1267. Naval Ocean Systems Center, San Diego, CA.
- Maiers J. and Y.S. Sherif. 1985. "Applications of Fuzzy Set Theory," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 15, no. 1.
- Miller, D. S. and M.K. Starr. 1969. *Executive Decisions and Operations Research*, Prentice-Hall, Englewood Cliffs, New Jersey.

- Miller, G.A. 1956. "The Magical Number Seven Plus or Minus Two: Some Limits on our Capacity for Processing Information," *The Psychological Review* (March), pp. 81-97 [referenced by Saaty].
- O'Hagan, M. 1987. "Fuzzy Decision Aids," *Conf. Record, 21st Asilomar Conf. on Circuits, Systems, and Computers*, Pacific Grove, CA.
- Saaty, T. L. 1972. *An Eigenvalue Allocation Model in Contingency Planning*, The Univ. of Pennsylvania, Philadelphia, PA.
- Saaty, T. L. 1977. "A Scaling Method for Priorities in Hierarchical Structures," *J. of Mathematical Psychology*, vol. 15, pp. 234-281.
- Schafer, G. 1976. *A Mathematical Theory of Evidence*, Princeton University Press, Princeton, NJ.
- Shortliffe, E.H. 1976. *Computer-Based Medical Consultations: MYCIN*. American Elsevier, New York.
- Starr, M. K. and M. Zeleny. 1977. *Multiple Criteria Decision Making*, Academic Press, New York.
- Yager, R. R. 1977. "Multiple Objective Decision-Making Using Fuzzy Sets," *Int. J. Man-Machine Studies*, vol. 9, pp. 375-382.
- Yager, R. R. 1980a. "A Foundation for a Theory of Possibility," *Journal of Cybernetics*, vol. 10, pp. 177-204.
- Yager, R. R. 1980b. "Fuzzy Decision Making Including Unequal Objectives," *Computers and Operations Research*, vol. 7, pp. 285-300.
- Yager, R. R. 1981. "A New Methodology for Ordinal Multiobjective Decisions Based on Fuzzy Sets," *Journal for the American Institute for Decision Sciences*, vol. 12, no. 4, pp. 589-600.
- Yager, R.R. 1987. *On Ordered Weighted Averaging Aggregation Operators in Multi-Criteria Decision Making*, Tech. Report #M11-705, Machine Intelligence Institute, Iona College, New Rochelle, New York.
- Yager, R. R. and D. Basson. 1975. "Decision-making with Fuzzy Sets," *Decision Sciences*, vol. 6, pp. 590-600.
- Zadeh, L. A. 1965. "Fuzzy Sets," *Information and Control*, vol. 8, pp. 338-353.
- Zadeh, L. A. and R. E. Bellman. 1970. "Decision-Making in a Fuzzy Environment," *Management Science*, vol. 17, pp. 141-164.
- Zidek, J. Y. 1987. "Group Decision Analysis and its Application to Combining Opinions," *Conf. Proc., Workshop on Assessing Uncertainty*, Naval Postgraduate School, Monterey, CA, Nov 1986. Dept. of Statistics, Stanford Univ.

Zimmerman, H. J. and P. Zysno. 1980. "Latent Connectives in Human Decision Making," *FSS*, vol. 4, pp. 37-51.

Zoinks, S. 1978. *Multiple Criteria Problem Solving*. Springer-Verlag, New York.

APPENDIX A:
PROGRAMMED EXAMPLES

The following examples were generated on a VAX computer using the fuzzy logic decision tool described in this report. These examples have been generated to coincide with examples in the text. The user-supplied inputs are shown in bold type.

The program that generated the examples is documented in appendices B and C. A source-code listing for this program is not included because of its length. A copy of the complete program can be made available upon request to the authors. The most useful components of the fuzzy logic decision tool are contained in appendices D and E. These listings are stand-alone source code for the Min-Max Pair-Wise Method and the Min-Max Rank Method, respectively. These short versions of the fuzzy logic algorithms do not include the weighted-sum method nor the explanation facility.

All code has been written in FranzLISP, which runs in a UNIX environment. Versions are also available in CommonLISP and InterLISP.

Script started on Wed Nov 30 13:19:10 1988

```
% lisp
Franz Lisp, Opus 38.92
-> (load 'fuzzy.l)
[load fuzzy.l]
```

* FUZZY LOGIC DECISION-MAKING TOOL *

- 1: Min-Max Rank Method
- 2: Min-Max Pair-Wise Method
- 3: Weighted-Sum Rank Method
- 4: Weighted-Sum Pair-Wise Method
- 5: Brief Description of Min-Max Rank Method
- 6: Brief Description of Min-Max Pair-Wise Method
- 7: Brief Description of Weighted-Sum Rank Method
- 8: Brief Description of the Weighted-Sum Pair-Wise Method
- 9: Exit

Which do you prefer ?

Enter 1, 2, 3, 4, 5, 6, 7, 8, or 9 ---> 1

* FUZZY LOGIC DECISION-MAKING TOOL *

A program based on R.Yager's 1981 paper:
"A New Methodology for Ordinal Multi-objective
Decisions Based on Fuzzy Sets"

ENTER NUMBER OF ALTERNATIVES: 3

ENTER LABEL FOR ALTERNATIVE 1: **Stop**
ENTER LABEL FOR ALTERNATIVE 2: **Turn**
ENTER LABEL FOR ALTERNATIVE 3: **No Change**

ENTER NUMBER OF CRITERIONS: 4

ENTER LABEL FOR CRITERION 1: **Avoid Collision**
ENTER LABEL FOR CRITERION 2: **Avoid Detection**
ENTER LABEL FOR CRITERION 3: **Keep Contact**
ENTER LABEL FOR CRITERION 4: **Keep Station**

	PREFERENCE MEASURES:	
	0. lowest or none	
	1. very low	
	2. low	
	3. medium	
	4. high	
	5. very high	
	6. perfect or absolute	

For Avoid Collision, enter rating for Stop (0 - 6): 3
 For Avoid Collision, enter rating for Turn (0 - 6): 4
 For Avoid Collision, enter rating for No Change (0 - 6): 2
 For Avoid Detection, enter rating for Stop (0 - 6): 3
 For Avoid Detection, enter rating for Turn (0 - 6): 2
 For Avoid Detection, enter rating for No Change (0 - 6): 5
 For Keep Contact, enter rating for Stop (0 - 6): 1
 For Keep Contact, enter rating for Turn (0 - 6): 0
 For Keep Contact, enter rating for No Change (0 - 6): 4
 For Keep Station, enter rating for Stop (0 - 6): 4
 For Keep Station, enter rating for Turn (0 - 6): 2
 For Keep Station, enter rating for No Change (0 - 6): 6

Enter rating for Avoid Collision (0 - 6): 5
 Enter rating for Avoid Detection (0 - 6): 1
 Enter rating for Keep Contact (0 - 6): 3
 Enter rating for Keep Station (0 - 6): 6

Our decision based on this model is Stop.

To see Explanation for the decision, enter E,
 To see your Input, enter I,
 To see Menu again, enter M,
 To Quit, enter Q,

Input E, I, M or Q ----> E

EXPLANATION

The No Change alternative is a better choice than Stop in satisfying the Avoid

Detection, Keep Contact and Keep Station criteria. But, since Stop satisfies the Avoid Collision criterion more closely than No Change, Stop is the better choice.

The Turn alternative is rejected partially because of a low rating in the Keep Station criterion. The No Change alternative is rejected partially because of a low rating in the Avoid Collision criterion.

Input E, I, M or Q ----> I

	Avoid Collision	Avoid Detection	Keep Contact	Keep Station
Stop	3	3	1	4
Turn	4	2	0	2
No Change	2	5	4	6
RATINGS FOR CRITERIA:				
	5	1	3	6

Input E, I, M or Q ----> M

Do you wish to use the Weighted-Sum Rank Method with the input from the Min-Max Rank Method ?

1. Yes
2. No

Please enter 1 or 2 ---> 1

	Avoid Collision	Avoid Detection	Keep Contact	Keep Station
Stop	3	3	1	4
Turn	4	2	0	2
No Change	2	5	4	6
RATINGS FOR CRITERIA:				
	5	1	3	6

EXPLANATION

Keep Station and Avoid Collision are 2 of the most important criteria, and since the No Change alternative satisfies Keep Station AND Avoid Collision best, No Change is the best choice.

Input E, I, M, or Q ----> M

*** FUZZY LOGIC DECISION-MAKING TOOL ***

- 1: Min-Max Rank Method
- 2: Min-Max Pair-Wise Method
- 3: Weighted-Sum Rank Method
- 4: Weighted-Sum Pair-Wise Method
- 5: Brief Description of Min-Max Rank Method
- 6: Brief Description of Min-Max Pair-Wise Method
- 7: Brief Description of Weighted-Sum Rank Method
- 8: Brief Description of the Weighted-Sum Pair-Wise Method
- 9: Exit

Which do you prefer ?

Enter 1, 2, 3, 4, 5, 6, 7, 8, or 9 ---> 2

*** FUZZY LOGIC DECISION-MAKING TOOL ***

A program based on R.Yager's 1977 paper:
"Multiple Object Decision-Making Using Fuzzy Sets"

ENTER NUMBER OF ALTERNATIVES: 3

ENTER LABEL FOR ALTERNATIVE 1: **Stop**
ENTER LABEL FOR ALTERNATIVE 2: **Turn**
ENTER LABEL FOR ALTERNATIVE 3: **No Change**

ENTER NUMBER OF CRITERIONS: 4

ENTER LABEL FOR CRITERION 1: **Avoid Collision**
ENTER LABEL FOR CRITERION 2: **Avoid Detection**
ENTER LABEL FOR CRITERION 3: **Keep Contact**
ENTER LABEL FOR CRITERION 4: **Keep Station**

RATING THE ALTERNATIVES:

Enter a value between 0 and 10; where a higher value indicates a better rating.

For Avoid Collision, enter rating for Stop: 5
For Avoid Collision, enter rating for Turn: 7
For Avoid Collision, enter rating for No Change: 3
For Avoid Detection, enter rating for Stop: 5
For Avoid Detection, enter rating for Turn: 4
For Avoid Detection, enter rating for No Change: 8
For Keep Contact, enter rating for Stop: 2
For Keep Contact, enter rating for Turn: 0.1
For Keep Contact, enter rating for No Change: 6
For Keep Station, enter rating for Stop: 6
For Keep Station, enter rating for Turn: 4
For Keep Station, enter rating for No Change: 9

RATING THE CRITERIA

To specify the degree of importance, enter a number between 1 and 9 where the values are:

- 1 - Equal Importance
- 3 - Weak Importance of One Over the Other
- 5 - Strong Importance of One Over the Other
- 7 - Demonstrated Importance of One Over the Other
- 9 - Absolute Importance of One Over the Other

Use 2, 4, 6 & 8 when the degree of importance falls between the values above.

- 1. Avoid Collision
- 2. Avoid Detection

Which of the criteria is more important? 1

By what degree? (Use scale above) 5

- 1. Avoid Collision
- 2. Keep Contact

Which of the criteria is more important? 1

By what degree? (Use scale above) 3

- 1. Avoid Collision
- 2. Keep Station

Which of the criteria is more important? 1

By what degree? (Use scale above) 1

- 1. Avoid Detection
- 2. Keep Contact

Which of the criteria is more important? 2

By what degree? (Use scale above) 7

1. Avoid Detection
2. Keep Station

Which of the criteria is more important? 2

By what degree? (Use scale above) 7

1. Keep Contact
2. Keep Station

Which of the criteria is more important? 2

By what degree? (Use scale above) 3

* decision ratings range from 0 to 10 *

* 0 ----> very poor *

* 10 ----> excellent *

RANKED DECISION LIST

	alternative	quantitative decision-value	subjective decision-value
	-----	-----	-----
1)	Stop	3.0	fair
2)	No Change	1.7	poor
3)	Turn	0.3	very poor

Input E, I, M, or Q ----> E

EXPLANATION

The No Change alternative is better than Stop in satisfying the Avoid Detection, Keep Contact and Keep Station criteria. Stop is better than No Change in satisfying the Avoid Collision criterion. With consideration to the overall importance of Avoid Collision versus Avoid Detection, Keep Contact and Keep Station, the degree of superiority of Stop over No Change in Avoid Collision is deemed to be greater than the degree of superiority of No Change over Stop in Avoid Detection, Keep Contact and Keep Station. Hence, Stop is preferred over No Change.

Input E, I, M, or Q ----> I

	Avoid Collision	Avoid Detection	Keep Contact	Keep Station
Stop	5.0	5.0	2.0	6.0
Turn	7.0	4.0	0.1	4.0
No Change	3.0	8.0	6.0	9.0
RATINGS FOR CRITERIA:				
	1.5	0.2	0.8	1.6

Input E, I, M, or Q ----> M

Do you wish to use the Weighted-Sum Pair-Wise Method with the input from the Min-Max Pair-Wise Method ?

1. Yes

2. No

Please enter 1 or 2 ---> 1

	Avoid Collision	Avoid Detection	Keep Contact	Keep Station
Stop	5.0	5.0	2.0	6.0
Turn	7.0	4.0	0.1	4.0
No Change	3.0	8.0	6.0	9.0
RATINGS FOR CRITERIA:				
	1.5	0.2	0.8	1.6

EXPLANATION

Keep Station and Avoid Collision are 2 of the most important criteria, and since the No Change alternative satisfies Keep Station AND Avoid Collision best, No Change is the best choice.

Input E, I, M, or Q ----> M

* FUZZY LOGIC DECISION-MAKING TOOL *

- 1: Min-Max Rank Method
- 2: Min-Max Pair-Wise Method
- 3: Weighted-Sum Rank Method
- 4: Weighted-Sum Pair-Wise Method
- 5: Brief Description of Min-Max Rank Method
- 6: Brief Description of Min-Max Pair-Wise Method
- 7: Brief Description of Weighted-Sum Rank Method
- 8: Brief Description of the Weighted-Sum Pair-Wise Method
- 9: Exit

Which do you prefer ?

Enter 1, 2, 3, 4, 5, 6, 7, 8, or 9 ---> **5**

**** DECISION MAKING TOOL USING THE MIN-MAX RANK METHOD ****

Multi-objective decision making is a process of selecting a decision from a set of decision alternatives, given a set of objectives with differing importance to the decision maker. The Min-Max Rank Method is a 'conservative' decision algorithm in which it is based on worst case analysis.

For each decision alternative, the Min-Max Rank Method finds the objective in which the alternative has the minimum rating, and then assigns that rating to the alternative as its 'overall' rating. Then, the Method selects the best decision alternative with the maximum 'overall' rating.

The Min-Max Rank Method requires the user to specify:

1. The set of alternatives
2. The set of objectives
3. The degree (from a scale 0 to 6) to which each alternative satisfies the objectives
4. The ranking (from a scale 0 to 6) of each objective

Please enter E or M ---> **M**

*** FUZZY LOGIC DECISION-MAKING TOOL ***

- 1: Min-Max Rank Method

- 2: Min-Max Pair-Wise Method
- 3: Weighted-Sum Rank Method
- 4: Weighted-Sum Pair-Wise Method
- 5: Brief Description of Min-Max Rank Method
- 6: Brief Description of Min-Max Pair-Wise Method
- 7: Brief Description of Weighted-Sum Rank Method
- 8: Brief Description of the Weighted-Sum Pair-Wise Method
- 9: Exit

Which do you prefer ?

Enter 1, 2, 3, 4, 5, 6, 7, 8, or 9 ---> 6

** DECISION-AID TOOL USING PAIR-WISE MIN-MAX METHOD **

Multi-objective decision making is a process of selecting a decision from a set of decision alternatives, given a set of objectives with differing importance to the decision maker. The Pair-Wise Min-Max Method differs from the Min-Max Rank Method in only one regard; it only requires pair-wise comparisons of objectives rather than the complete ranking of all the objectives.

The Pair-Wise Min-Max Method requires the user to specify :

- 1. The set of alternatives
- 2. The set of objectives
- 3. The degree (from a scale 0 to 10) to which each alternative satisfies the objectives
- 4. The pair-wise comparisons of importance of objectives

Please enter E or M ---> M

* FUZZY LOGIC DECISION-MAKING TOOL *

- 1: Min-Max Rank Method
- 2: Min-Max Pair-Wise Method
- 3: Weighted-Sum Rank Method
- 4: Weighted-Sum Pair-Wise Method
- 5: Brief Description of Min-Max Rank Method
- 6: Brief Description of Min-Max Pair-Wise Method
- 7: Brief Description of Weighted-Sum Rank Method
- 8: Brief Description of the Weighted-Sum Pair-Wise Method
- 9: Exit

Which do you prefer ?

Enter 1, 2, 3, 4, 5, 6, 7, 8, or 9 ---> 7

*DECISION MAKING TOOL USING THE WEIGHTED-SUM RANK METHOD

Multi-objective decision making is a process of selecting a decision from a set of decision alternatives, given a set of objectives with differing importance to the decision maker. The Weighted-Sum Rank Method is an 'optimistic' decision algorithm based on average case analysis. The 'overall' rating of an alternative is the average degree to which the alternative satisfies the objectives, where each objective is weighted according to its importance. The Weighted-Sum Rank Method will select the alternative with the highest 'overall' rating.

The Weighted-Sum Rank Method requires the user to specify :

1. The set of alternatives
2. The set of objectives
3. The degree (from a scale 0 to 6) to which each alternative satisfies the objectives
4. The ranking (from a scale 0 to 6) of each objective

Please enter E or M ---> M

* FUZZY LOGIC DECISION-MAKING TOOL *

- 1: Min-Max Rank Method
- 2: Min-Max Pair-Wise Method
- 3: Weighted-Sum Rank Method
- 4: Weighted-Sum Pair-Wise Method
- 5: Brief Description of Min-Max Rank Method
- 6: Brief Description of Min-Max Pair-Wise Method
- 7: Brief Description of Weighted-Sum Rank Method
- 8: Brief Description of the Weighted-Sum Pair-Wise Method
- 9: Exit

Which do you prefer ?

Enter 1, 2, 3, 4, 5, 6, 7, 8, or 9 ---> 8

* DECISION MAKING TOOL USING THE WEIGHTED-SUM PAIR-WISE METHOD *

Multi-objective decision making is a process of selecting a decision from a set of decision alternatives, given a set of objectives with differing importance to the decision maker. The Weighted-Sum Pair-Wise Method is an 'optimistic' decision algorithm based on average or typical case analysis. The 'overall' rating of an alternative is the average degree to which the alternative satisfies the objectives, where each objective is weighted according to its importance. The Weighted-Sum Pair-Wise Method will select the alternative with the highest 'overall' rating.

The Weighted-Sum Pair-Wise Method requires the user to specify :

1. The set of alternatives
2. The set of objectives
3. The degree (from a scale 0 to 10) to which each alternative satisfies the objectives
4. The pair-wise comparisons of the objectives.

Please enter E or M ---> **M**

*** FUZZY LOGIC DECISION-MAKING TOOL ***

- 1: Min-Max Rank Method
- 2: Min-Max Pair-Wise Method
- 3: Weighted-Sum Rank Method
- 4: Weighted-Sum Pair-Wise Method
- 5: Brief Description of Min-Max Rank Method
- 6: Brief Description of Min-Max Pair-Wise Method
- 7: Brief Description of Weighted-Sum Rank Method
- 8: Brief Description of the Weighted-Sum Pair-Wise Method
- 9: Exit

Which do you prefer ?

Enter 1, 2, 3, 4, 5, 6, 7, 8, or 9 ---> **9**

script done on Wed Nov 30 13:34:18 1988

Script started on Thu Dec 1 12:03:01 1988
% lisp

Franz Lisp, Opus 38.92
-> (load 'fuzzy.l))
[load fuzzy.l]

*** FUZZY LOGIC DECISION-MAKING TOOL ***

- 1: Min-Max Rank Method
- 2: Min-Max Pair-Wise Method
- 3: Weighted-Sum Rank Method
- 4: Weighted-Sum Pair-Wise Method
- 5: Brief Description of Min-Max Rank Method
- 6: Brief Description of Min-Max Pair-Wise Method
- 7: Brief Description of Weighted-Sum Rank Method
- 8: Brief Description of the Weighted-Sum Pair-Wise Method
- 9: Exit

Which do you prefer ?

Enter 1, 2, 3, 4, 5, 6, 7, 8, or 9 ---> 2

*** A FUZZY LOGIC DECISION-MAKING TOOL ***

A program based on R.Yager's 1977 paper
"Multiple Object Decision-Making Using Fuzzy Sets"

ENTER NUMBER OF ALTERNATIVES: 3

ENTER LABEL FOR ALTERNATIVE 1: Stop
ENTER LABEL FOR ALTERNATIVE 2: Turn
ENTER LABEL FOR ALTERNATIVE 3: No Change

ENTER NUMBER OF CRITERIONS: 4

ENTER LABEL FOR CRITERION 1: Avoid Collision
ENTER LABEL FOR CRITERION 2: Avoid Detection
ENTER LABEL FOR CRITERION 3: Keep Contact
ENTER LABEL FOR CRITERION 4: Keep Station

RATING THE ALTERNATIVES:

Enter a value between 0 and 10; where a higher value indicates a better rating.

For Avoid Collision, enter rating for Stop: 5
For Avoid Collision, enter rating for Turn: 7
For Avoid Collision, enter rating for No Change: 3
For Avoid Detection, enter rating for Stop: 5
For Avoid Detection, enter rating for Turn: 4
For Avoid Detection, enter rating for No Change: 8
For Keep Contact, enter rating for Stop: 2
For Keep Contact, enter rating for Turn: 0.1
For Keep Contact, enter rating for No Change: 6
For Keep Station, enter rating for Stop: 6
For Keep Station, enter rating for Turn: 4
For Keep Station, enter rating for No Change: 9

RATING THE CRITERIA

To specify the degree of importance, enter a number between 1 and 9 where the values are:

- 1 - Equal Importance
- 3 - Weak Importance of One Over the Other
- 5 - Strong Importance of One Over the Other
- 7 - Demonstrated Importance of One Over the Other
- 9 - Absolute Importance of One Over the Other

Use 2, 4, 6 & 8 when the degree of importance falls between the values above.

- 1. Avoid Collision
- 2. Avoid Detection

Which of the criteria is more important? 2

By what degree? (Use scale above) 5

- 1. Avoid Collision
- 2. Keep Contact

Which of the criteria is more important? 1

By what degree? (Use scale above) 3

- 1. Avoid Collision
- 2. Keep Station

Which of the criteria is more important? 1

By what degree? (Use scale above) 1

- 1. Avoid Detection
- 2. Keep Contact

Which of the criteria is more important? 2

By what degree? (Use scale above) **5**

1. Avoid Detection

2. Keep Station

Which of the criteria is more important? **2**

By what degree? (Use scale above) **3**

1. Keep Contact

2. Keep Station

Which of the criteria is more important? **1**

By what degree? (Use scale above) **1**

PAIRED-COMPARISON OF CRITERIA

	1	2	3	4
1	1	1/5	3	1
2	5	1	1/5	1/3
3	1/3	5	1	1
4	1	3	1	1

1 = Avoid Collision

2 = Avoid Detection

3 = Keep Contact

4 = Keep Station

THERE IS INCONSISTENCY IN THE MATRIX OF PAIRED-COMPARISONS OF CRITERIA

THE FOLLOWING LIST OF INCONSISTENCIES IS NOT EXHAUSTIVE.

THE INCONSISTENCY IS AS FOLLOWS:

Keep Contact is more important than Avoid Detection by degree 5.0

Avoid Collision is more important than Keep Contact by degree 3.0

But Avoid Detection is more important than Avoid Collision by degree 5.0

THE INCONSISTENCY IS AS FOLLOWS:

Keep Station is more important than Avoid Detection by degree 3.0

Avoid Collision is more important than Keep Station by degree 1.0

But Avoid Detection is more important than Avoid Collision by degree 5.0

To minimize the inconsistency listed above, it is recommended that the following change be made:

Avoid Collision is more important than Avoid Detection with degree 7

(1, 2) --> 7

Would you like to make this change? [(y)es/(n)o] y

* decision ratings range from 0 to 10 *

* 0 ----> very poor *

* 10 ----> excellent *

RANKED DECISION LIST

	alternative	quantitative decision-value	subjective decision-value
	-----	-----	-----
1)	Stop	2.9	fair
2)	No Change	1.8	poor
3)	Turn	0.1	very poor

Input E, I, M, or Q ---> q

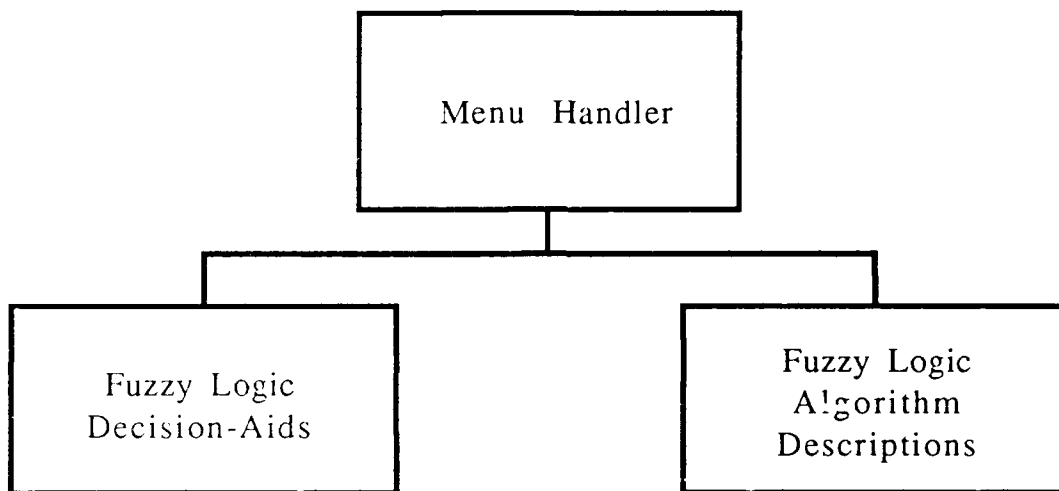
%

script done on Thu Dec 1 12:08:03 1988

APPENDIX B:
FUNCTIONAL DIAGRAMS

A FUZZY LOGIC DECISION-MAKING TOOL

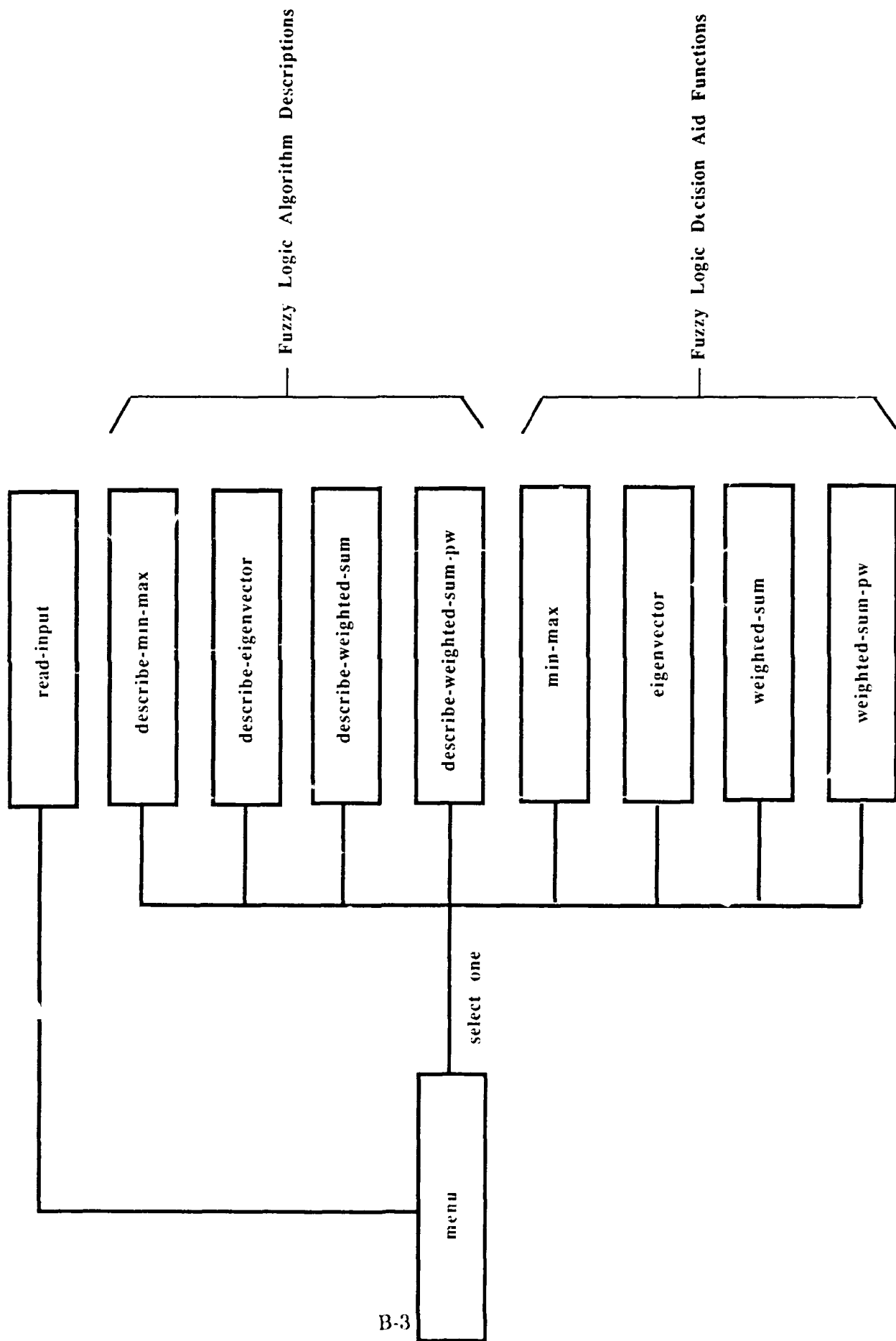
Functional Overview



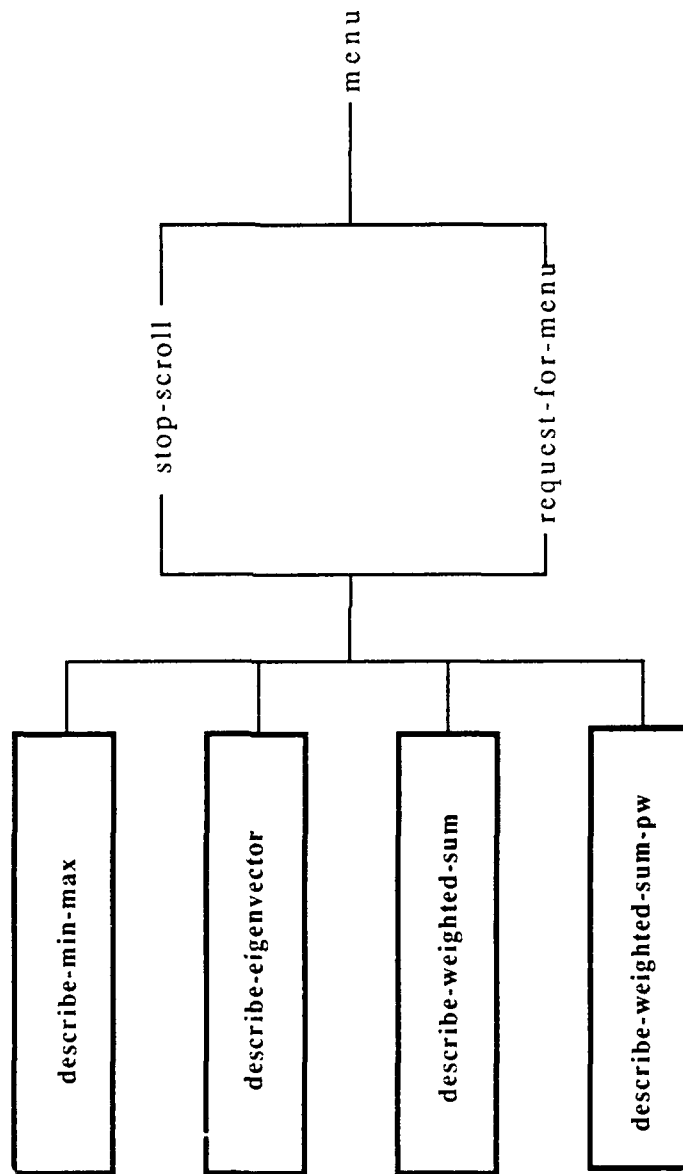
Fuzzy Logic Algorithms:

- Min-Max Method
- Eigenvector Method
- Weighted-Sum Method
- Weighted-Sum-PW Method

A FUZZY LOGIC DECISION-MAKING TOOL Control Flow Diagram



A FUZZY LOGIC DECISION-MAKING TOOL
Functional Hierarchy For Algorithm Descriptions



```

graph TD
    subgraph "init-min-max"
        direction TB
        A1[ ]
        A2[ ]
        A3[ ]
        A4[ ]
        A5[ ]
        A6[ ]
    end
    subgraph "find-min-max-decision"
        direction TB
        B1[ ]
        B2[ ]
        B3[ ]
        B4[ ]
    end
    A1 --- B1
    A2 --- B2
    A3 --- B3
    A4 --- B4
    A5 --- B4
    A6 --- B4

    A1 --> C1[create-min-max-header]
    A2 --> C2[get-labels]
    A3 --> C3[print-measures]
    A4 --> C4[min-max-get-alt-ratings]
    A5 --> C5[min-max-get-crit-ratings]
    A6 --> C6[copy-alt-ratings-2]
    A6 --> C7[sort-crit-list]

    B1 --> C8[adjust-ratings]
    B2 --> C9[copy-alt-ratings-3]
    B3 --> C10[min-max-compute-expected-best]
    B4 --> C11[min-max-make-decision]

    C2 --> D1[discrm]
    C2 --> D2[read-x-in-list]
    C10 --> D3[sum-of-row]
    C10 --> D4[less-than-7]
    C11 --> D5[min-max-make-decision-list]

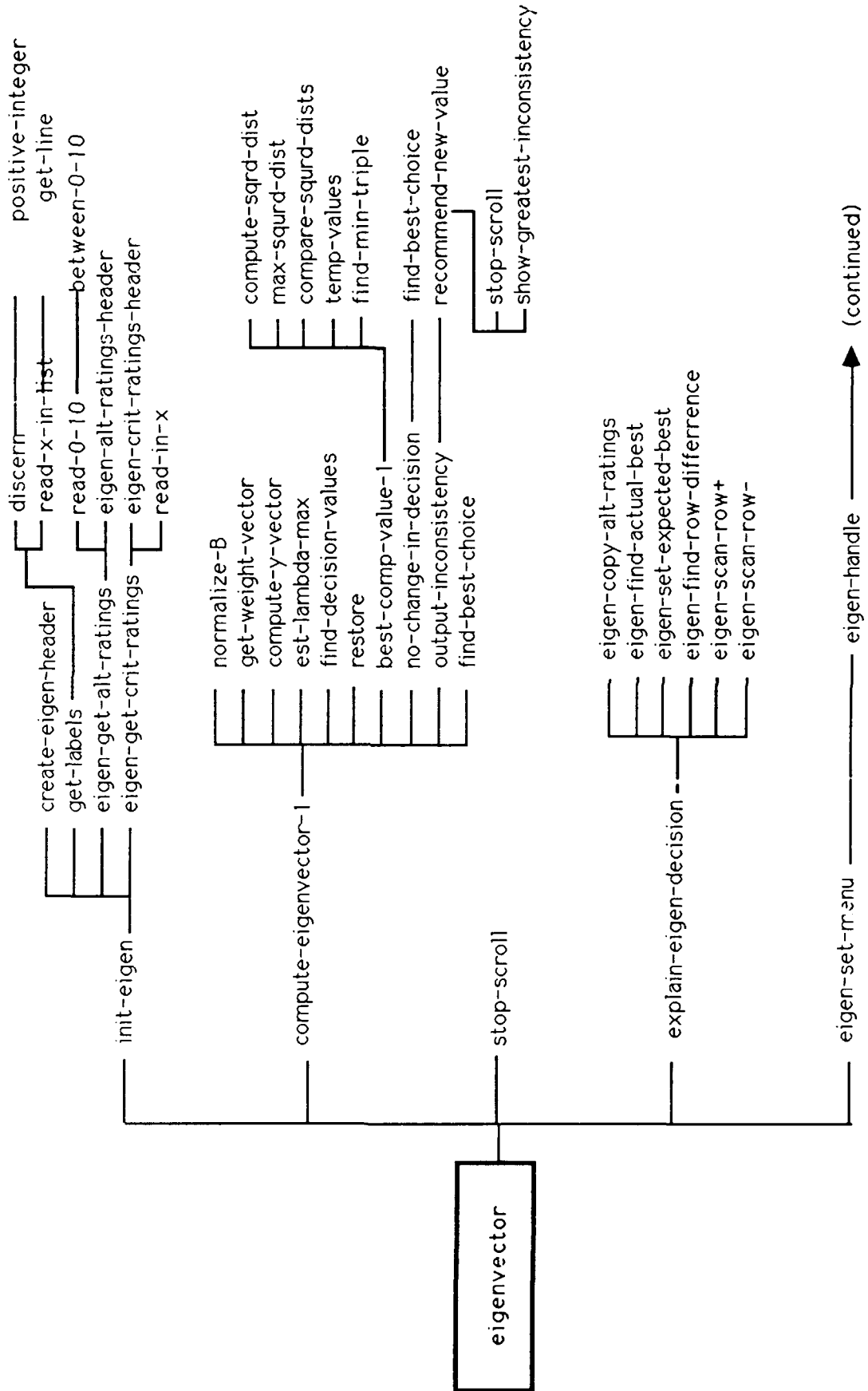
    D1 --> E1[positive-integer]
    D2 --> E2[get-line]

```

min-max-handle

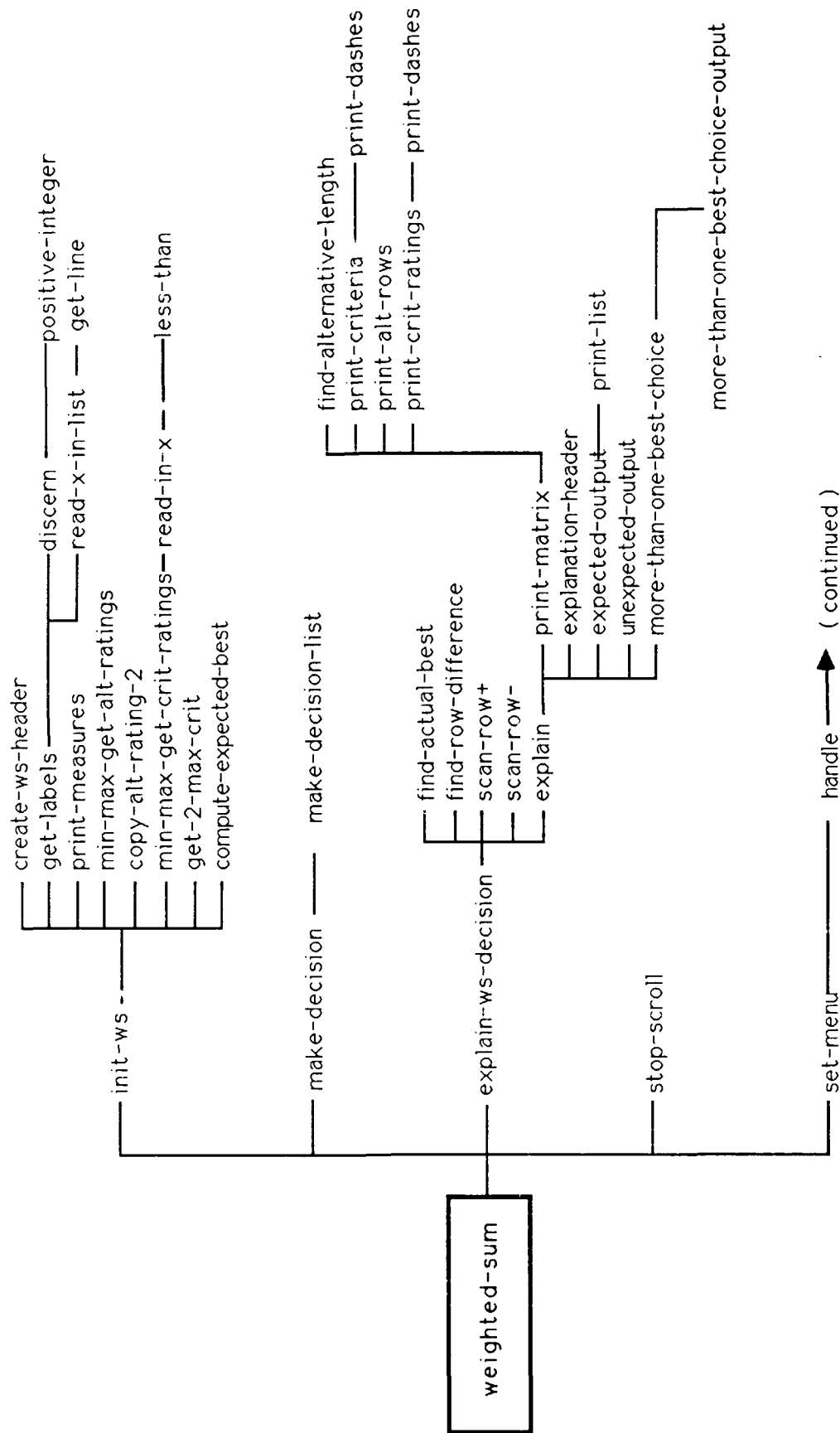
A FUZZY LOGIC DECISION-MAKING TOOL

Functional Hierarchy For Eigenvector

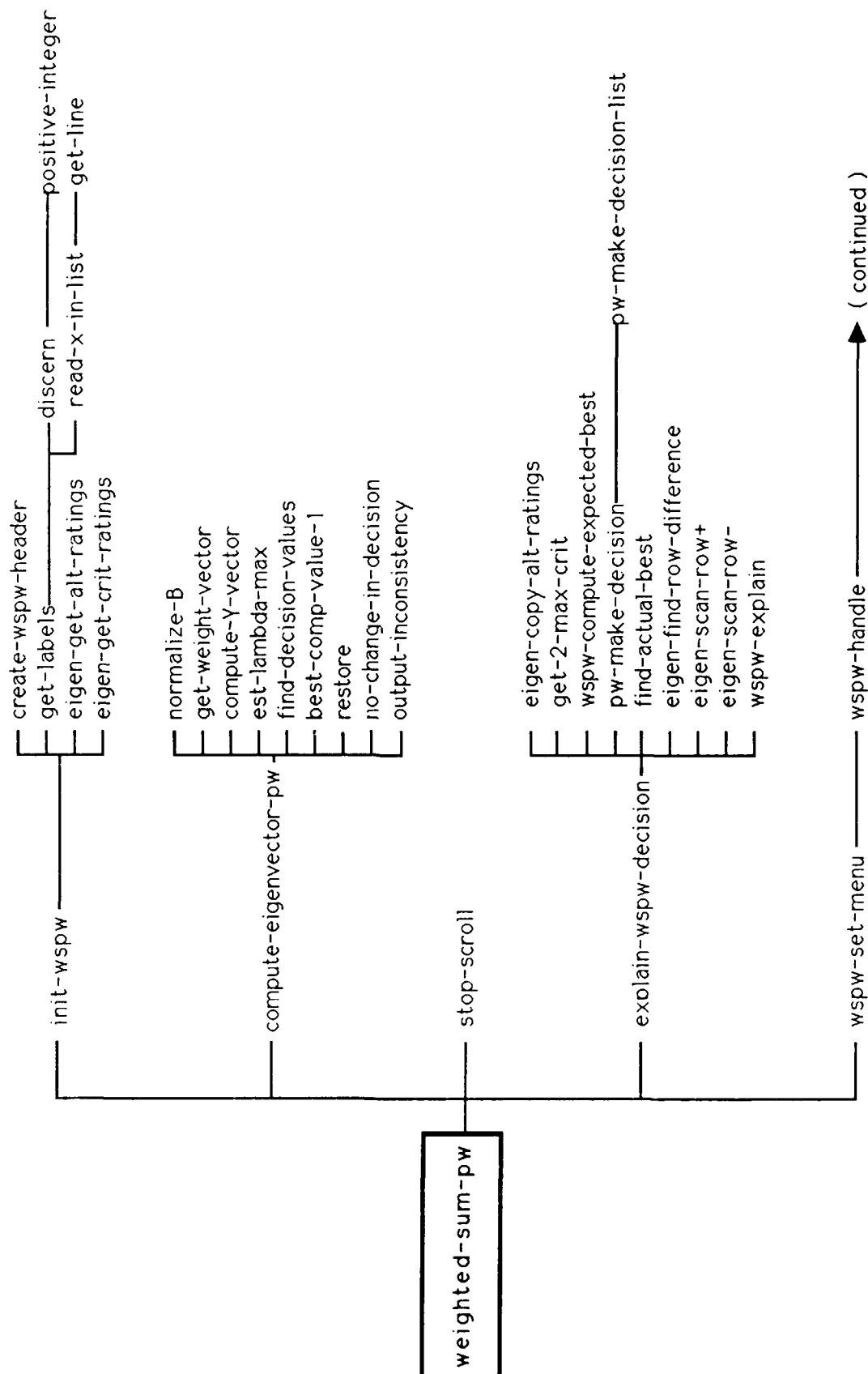


(continued)

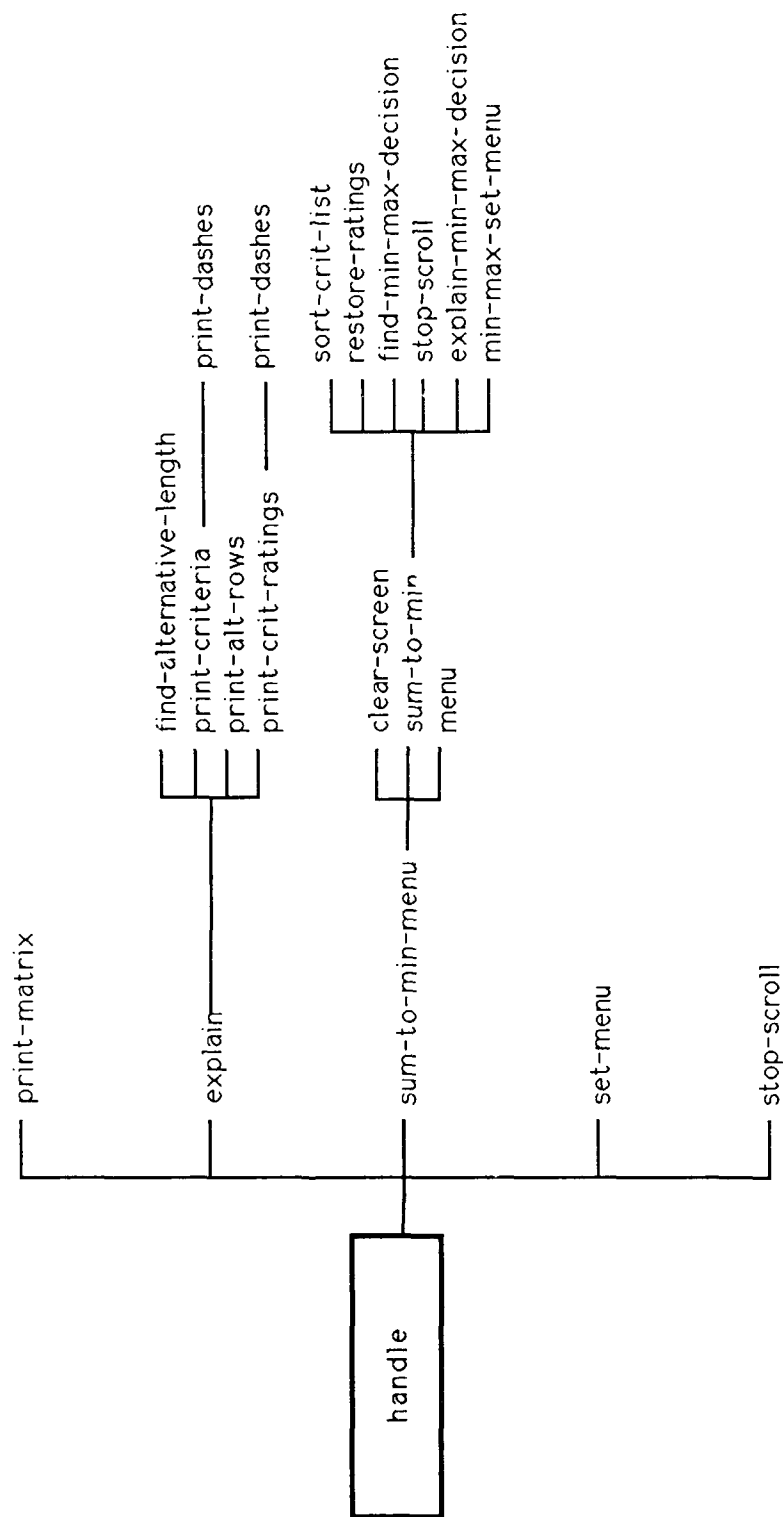
A FUZZY LOGIC DECISION-MAKING TOOL Functional Hierarchy For Weighted-Sum



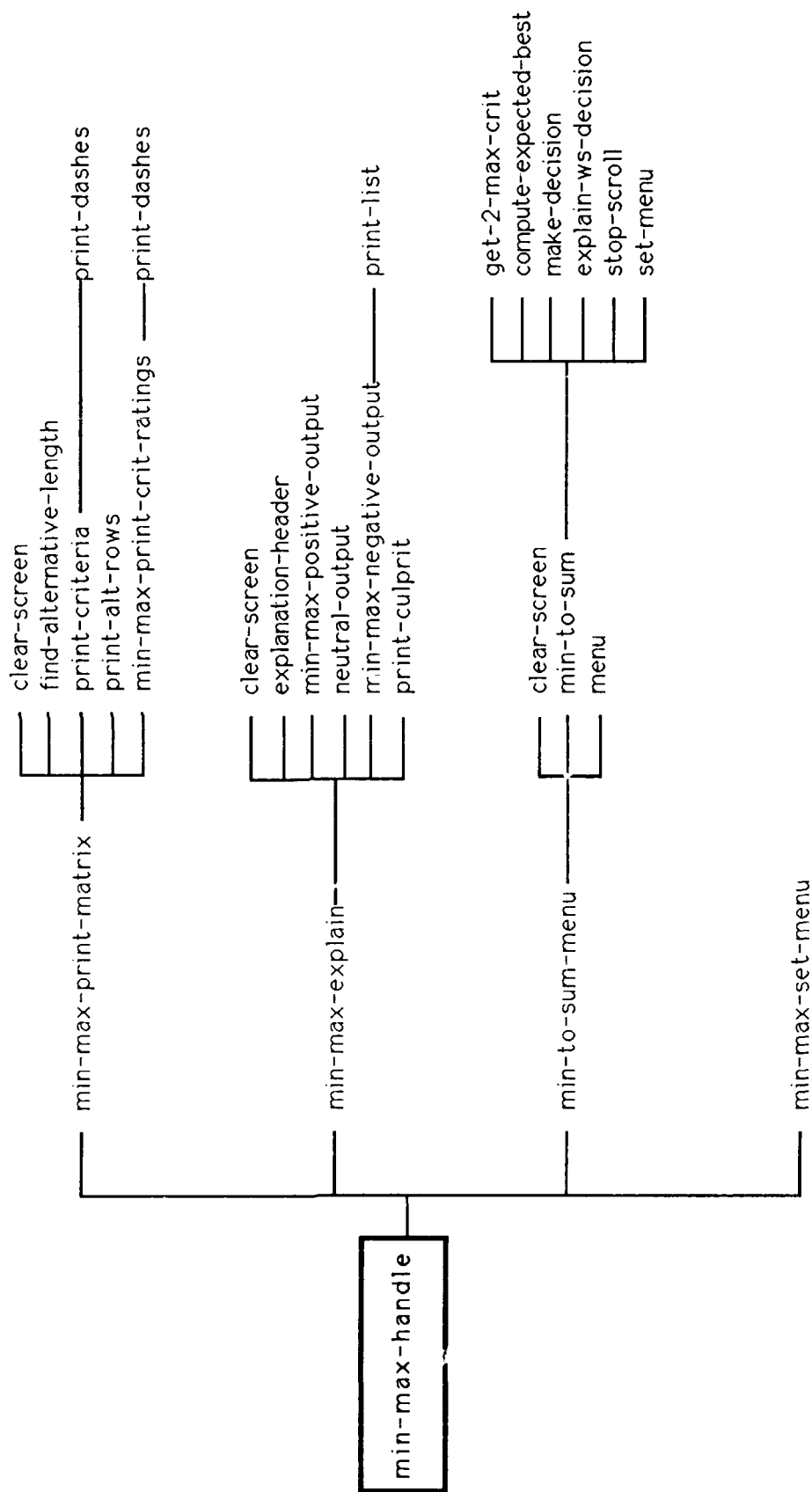
A FUZZY LOGIC DECISION-MAKING TOOL Functional Hierarchy For Weighted-Sum-PW



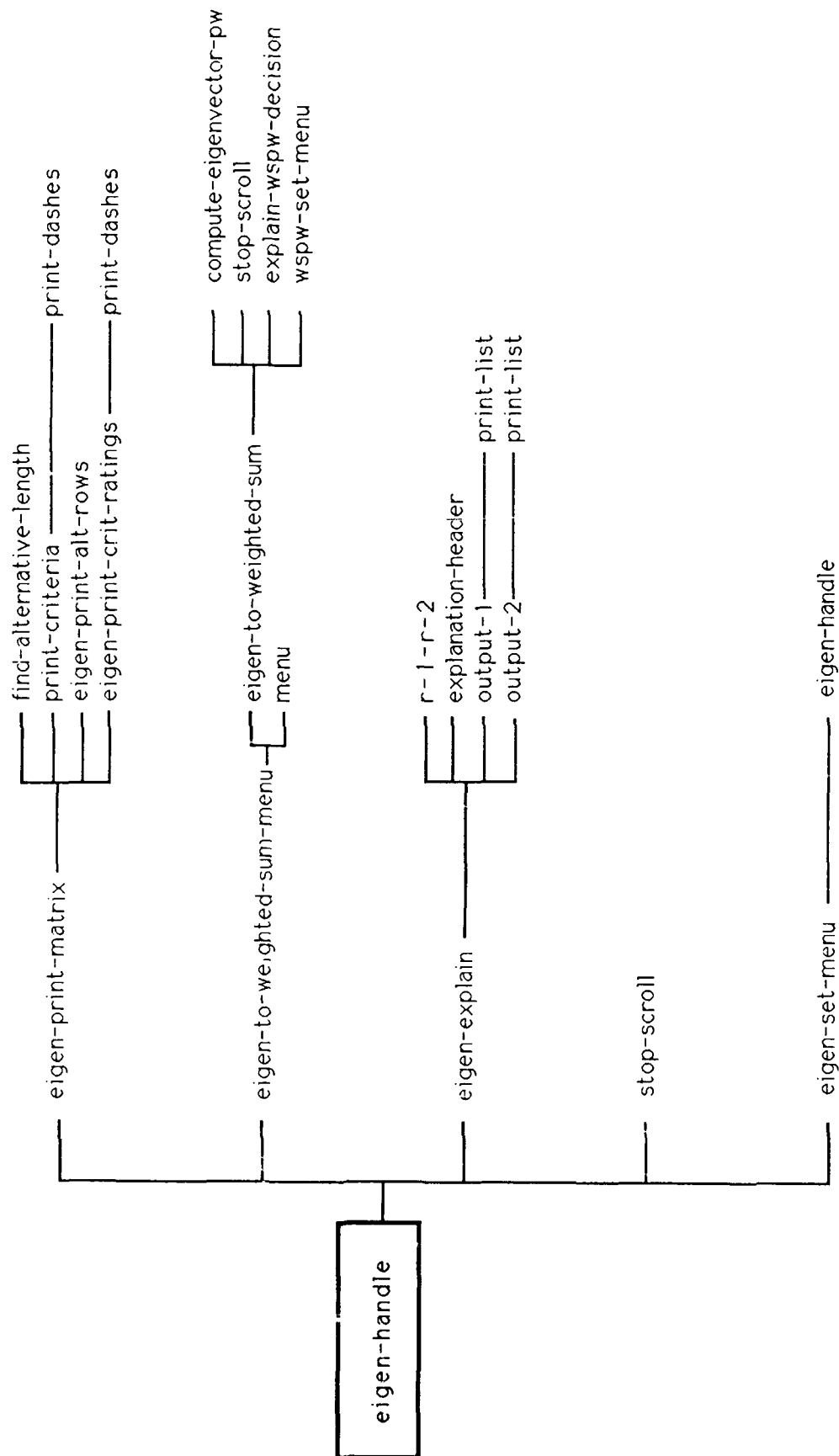
A FUZZY LOGIC DECISION-MAKING TOOL Functional Hierarchy For Handle



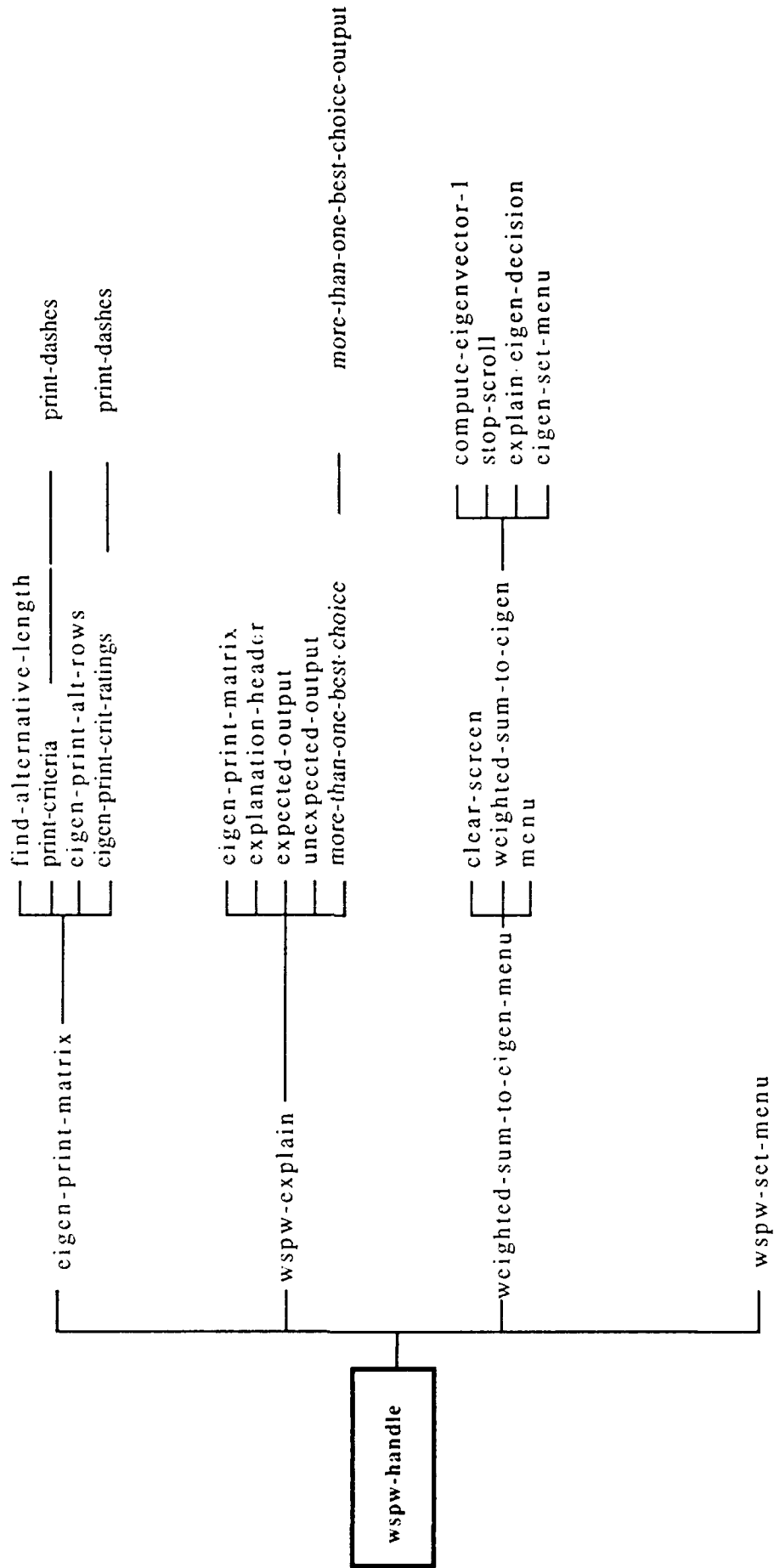
A FUZZY LOGIC DECISION-MAKING TOOL Functional Hierarchy For Min-Max-Handle



A FUZZY LOGIC DECISION-MAKING TOOL Functional Hierarchy For Eigen-Handle



A FUZZY LOGIC DECISION-MAKING TOOL Functional Hierarchy For WSPW-Handle



APPENDIX C:
FUNCTION DESCRIPTIONS

A FUZZY LOGIC DECISION-MAKING TOOL

Function Descriptions

adjust-ratings ()

Adjusts the ratings of the alternatives using $C_i = b_i \vee A_i$ (where C_i is the new rating, b_i is the adjusted criterion rating, and A_i is the actual alternative rating).

best-comp-value-1 ()

Finds the two criteria that cause the most inconsistencies and proceeds to change the relative importance of the two criteria to reduce inconsistencies.

between-0-10 ()

Will determine whether x is a number between 0 and 10.

clear-screen ()

Clears the display screen.

compare-squard-dists ()

Compares Max squared distance against the others and finds the next largest $D(j)$ value.

compute-expected-best ()

Finds the most intuitively likely alternative and puts its index into expected best; in case of ties, we will always choose the one that is nearest to the top of the alternative list as expected-best.

compute-eigenvector-1 ()

Finds the largest eigenvalue and its eigenvector of matrix B . If there are inconsistencies in B , the function will modify B to reduce inconsistencies. If the outcome of the modified B and the original B is the same, compute-eigenvector-1 will exit; otherwise, it will advise the user of the inconsistencies. This function uses an approximation to calculate the eigenvalues:

1. Normalize each column of B to yield B' .
2. Take the average over the resulting rows, yielding the weight vector W .
3. Compute BW and divide each of the components of the resulting vector by the corresponding component of W . This yields the Y vector.
4. Averaging the Y vector components gives the estimate of the maximum eigenvalue. (λ -max).

compute-eigenvector-pw ()

Finds the largest eigenvalue and its eigenvector of matrix B. If there are inconsistencies in B, the function will modify B to reduce inconsistencies. If the outcome of the modified B and the original B is the same, compute-eigenvector-pw will exit; otherwise, it will advise the user of the inconsistencies.

compute-expected-best ()

Finds the alternative that best satisfies the two most important criteria. This is determined by computing the highest value for $\text{alt-ratings}(c1,i) * \text{crit-ratings}(c1) + \text{alt-ratings}(c2,i) * \text{crit-ratings}(c2)$, where $c1$ = index of most important criteria, $c2$ = index of second most important criteria, i = alternative index.

compute-squrd-dist ()

Computes the squared distance for each column (jth) of B'.

$$D(j) = (B'(1, j) - w(1))^{**2} + (B'(2, j) - w(2))^{**2} + \dots$$

compute-Y-vector ()

Computes $B*W$ and divides each of the components of the resulting vector by the corresponding component of W to yield the Y vector.

copy-alt-ratings-2 ()

Makes a copy of the matrix alt-ratings into alt-ratings-2.

copy-alt-ratings-3 ()

Makes a copy of the input matrix, scaled by the weights of the criteria.

create-min-max-header ()

Displays the program header.

create-eigen-header ()

Displays the program title.

create-ws-header ()

Prints out the header for the Weighted-Sum Method.

create-wspw-header ()

Displays the header for the Weighted-Sum-Pair-Wise Method.

describe-eigenvector ()

Gives a brief discussion of the Min-Max Pair-Wise Method.

describe-min-max ()

Gives a brief description of the Min-Max Rank Model.

describe-weighted-sum ()

Gives a brief discussion of the Weighted-Sum Model.

describe-weighted-sum-pw ()

Output a brief description of the Weighted-Sum Pair-Wise Method.

discern ()

Will prompt the user until he/she enters a positive integer.

eigen-alt-ratings-header ()

Displays a header for degrees of preference.

eigen-copy-alt-ratings ()

Makes a copy of the matrix alt-ratings into alt-ratings-2.

eigen-crit-ratings-header ()

Displays the header for the RATING THE CRITERIA screen.

eigen-explain ()

Determines whether expected-best = actual-best and outputs the explanation accordingly.

eigen-find-actual-best ()

Puts the index of the best alternative into actual-best.

eigen-find-row-difference ()

Computes the difference between the expected-best and actual-best alternatives.

eigen-get-alt-ratings ()

Queries user to rate each alternative based on each criterion.

eigen-get-crit-ratings ()

Prompts the user to do pair-wise comparisons between all criteria in order to determine their relative importance.

eigen-handle ()

Performs the appropriate tasks depending on the input x; if x = q, then program exits; x = e, prints explanation; x = i, prints input; x = m, menu.

eigen-print-alt-rows ()

Prints the list of alternatives and the degrees to which they satisfy various criteria.

eigen-print-crit-ratings ()

Prints the weights of the criteria.

eigen-print-matrix ()

Prints the inputting matrix.

eigen-scan-row+ ()

Puts the indices corresponding to those criteria to which the actual-best alternative is better than the expected-best alternative.

eigen-scan-row- ()

Puts the indices corresponding to those criteria to which the actual-best alternative is better than the expected-best alternative.

eigen-set-expected-best ()

Set to the second-best alternatives on the decision-list.

eigen-set-menu ()

Prompts the user for the next input. If the next input is 'E', the program will display explanation; 'I', input; 'Q', quit; 'M', menu.

eigen-to-weighted-sum ()

Runs the Weighted-Sum Pair-Wise Method using the input from the previous run of the Min-Max Pair-Wise Method.

eigen-to-weighted-sum-menu ()

Is activated when the Min-Max Pair-Wise Method has finished its run. The menu asks the user whether he/she wants to use the Weighted-Sum Pair-Wise Method using the input from the previous running of the Min-Max Pair-Wise Method.

eigenvector ()

Drives the Min-Max Pair-Wise Algorithm.

est-lambda-max ()

Approximates the maximum eigenvalue by averaging the components of the Y vector.

expected-output ()

Prints the explanation when the expected-best alternative is also the actual-best alternative.

explain ()

Determines whether expected-best = actual-best and outputs the explanation accordingly.

explain-eigen-decision ()

Collects pertinent data for creating an explanation for the decision value.

explain-min-max-decision ()

Creates an explanation for the decision by comparing the min-max decision versus the decision that the user may have expected.

explain-ws-decision ()

Creates an explanation for the program's decision. This is done by determining what the "expected" choice would be, then comparing that choice with the program's choice.

explain-wspw-decision ()

Creates the explanation.

explanation-header ()

Simply prints the header.

find-actual-best ()

Puts the index of the best alternative into actual-best.

find-alternative-length ()

Finds the maximum number of characters that is in any of the alternative labels and places it in the variable "max-length".

find-best-choice ()

Displays to the user the ranked list of alternatives.

find-decision-value ()

Computes a decision-value for each i,

$\min(\text{alt-rating}(i, j) \cdot [W(j) \cdot m])$ for all j.

This value gets put in the decision-list and is used in determining the best choice. The best choice will be the alternative corresponding to the maximum decision-value.

find-min-list ()

Keeps track of the criterion in which an alternative has the lowest rating.

find-min-max-decision ()

Executes the steps for determining the best alternative.

find-min-triple ()

Finds the value that minimizes the triples that have the two worst inconsistency values in common. For example, if it has been determined that B(1,2) is the most inconsistent component of the B matrix, then we are interested in minimizing the sum of

$$B(1,2)*B(2,3) / B(1,3) + B(1,2)*B(2,4) / B(1,4).$$

find-row-difference ()

Takes the difference between the ratings of the "actual" best choice and the ratings of the "expected" best choice. Scan-row + determines the criteria in which the "actual" rates better than the "expected." Scan-row- does a similar action except reversing the action. Finally, explain determines whether the "expected" = the "actual" and outputs an appropriate explanation. Find-actual-best, find-row-difference, scan-row + & scan-row- are functions from the Min-Max Program.

get-2-max-crit ()

Finds the two most important criteria.

get-labels ()

Gets labels for criteria and alternatives.

get-line ()

Reads the current line into buffer "line."

get-weight-vector ()

Takes the average over the B' rows.

handle ()

Performs the appropriate tasks depending on the input x; if x = q, then program quits; x = e, prints explanation; x = i, prints input; x = m, displays menu.

init-eigen ()

Prompts the user for the alternatives, criteria, and their ratings.

init-min-max ()

Initializes the program by creating the global variables and getting the decision parameters (alternatives & criteria) from the user.

init-ws ()

Initializes the program by prompting the user for the ratings and labels for the alternatives and criteria. Then using this input, init-ws() defines the global data structures.

init-wspw ()

Prompts the user for the input parameters (labels, quantity, and ratings for the alternatives and criteria) then initializes the global data structures.

less-than ()

Will determine whether x is an integer less than or equal to y and greater than z; z,y are expected to be integers.

less-than-7 ()

Checks for ratings less than 7.

make-decision ()

Finds the optimal solution x^* that satisfies $D(x^*) = \text{Max } D(x)$ for all x where x is an element of the set of alternatives.

make-decision-list ()

Creates a decision-list of the form ((alt-decision-value.alt-number)...). The decision value is computed by taking the sum of alt-ratings(i, j)*crit-ratings(i) for each j (alternative) over all i (criteria).

max-squid-dist ()

Finds maximum D(j) value.

menu ()

Activates the top-level menu for the decision aid tool. The decision-making tool has four different implementations, and the user can choose among the four.

min-max ()

Drives the Min-Max Rank Algorithm.

min-max-explain ()

Determines whether expected-best = actual-best and outputs the explanation accordingly.

min-max-get-alt-ratings ()

Queries user to rate each alternative based on each criterion.

min-max-get-crit-ratings ()

Queries user to rate each criterion.

min-max-make-decision ()

Finds the optimal solution x^* that satisfies $D(x^*) = \text{Max } D(x)$ for all x where x is an element of the set of alternatives.

min-max-make-decision-list (alt-list)

Creates the set $D = C_1 \wedge C_2 \wedge \dots \wedge C_p$ where $D(x) = \text{Min}[C_i(x)]$ and $C_i(x) = B_i \vee A_i(x)$.

min-max-positive-output ()

Prints the explanation when the expected-best alternative is also the actual-best alternative.

min-max-negative-output ()

Prints the explanation when the expected-best alternative is different from the actual-best alternative.

min-max-print-crit-ratings ()

Prints the weights of the criteria.

min-max-print-matrix ()

Prints the inputting matrix and the weights of the criteria.

min-max-set-menu ()

Prompts the user for the next input. If the next input is 'E', the program will display explanation; 'I', input; 'Q', quit; 'M', menu.

min-max-handle ()

Performs the appropriate tasks depending on the input x ; if $x = q$, then program resets; $x = e$, prints explanation; $x = i$, prints input; $x = m$, menu.

min-to-sum ()

Uses the same input from the Min-Max Rank Method as new input to the Weighted-Sum Rank Method.

min-to-sum-menu ()

This menu is activated after the user finishes using the Min-Max Rank Method. Since the inputting formats of the Min-Max Rank Method and the Weighted-Sum Rank Method are the same, the input of the Min-Max Rank Method can be used as the input of the Weighted-Sum Rank Method. The menu will ask the user whether he/she wants to try the

Weighted-Sum Rank Method using the input from the Min-Max Rank Method.

more-than-one-best-choice ()

Determines whether the program has produced more than one alternative with the highest rating; if yes, puts all best alternatives in best-choice list.

more-than-one-best-choice-output ()

Prints the explanation when there is more than one choice with the top rating.

neutral-output ()

Prints explanation when actual-best is not better than the expected-best in any of the criteria.

no-change-in-decision ()

Determines whether there is a change in the outcome if the input were modified; if yes, returns 't'; otherwise, returns 'nil'.

normalize-B ()

Normalizes each column of matrix B (matrix of criteria ratings) to yield B'.

output-1 ()

Outputs explanation for a decision where the top two alternatives have identical ratings.

output-2 ()

Outputs part of the explanation for the final decision when the top choice has a higher rating than the second-best choice.

output-inconsistency ()

Advises the user of the inconsistencies of input.

positive-integer ()

Will determine whether an input x is a positive integer; if yes, returns 'x'; otherwise returns 'nil'.

print-alt-rows ()

Prints the ratings of an alternative with respect to criteria.

print-criteria ()

Prints the list of criteria.

print-crit-ratings ()

Prints the weights of the criteria.

print-culprit ()

Prints the criterion that is partially responsible for an alternative being rejected as the best choice.

print-dashes ()

Prints a line of dashes; the length of the line depends on the size of input.

print-list ()

Prints a sublist of the list "criteria."

print-matrix ()

Prints the inputting matrix and the weights of the criteria.

print-measures ()

Displays preference measures menu.

pw-make-decision ()

Is used for the Weighted-Sum Pair-Wise Method.

pw-make-decision-list ()

Given alt-list, which is a list of alternatives' indices, pw-make-decision-list creates a "decision list" containing the decision value of $\text{alt-rating}(i, j) * \text{crit-rating}(j)$ for each alternative.

The decision has the format:

((< decision value > . < alternative index >)...).

r-1-r-2 ()

Copies the two highest ratings on the decision-list into rating-1 and rating-2, accordingly.

read-0-10 ()

Will prompt the user to input a number between 0 and 10; if the user inputs something other than such number, the function will prompt the user again to input a "valid" number. The function returns the value of valid input.

read-input ()

Reads input. If input is 1, 2, 3, or 4, it returns input as the value of read-input; otherwise, it prompts the user for a valid input of 1, 2, 3, or 4.

read-in-x ()

Will return the value of x when x is an integer between the integers y and z; otherwise, the function will prompt the user for a "valid" input x.

read-x-in-list ()

Will read an input x; it will check whether x is an element of l; if yes, take appropriate actions; otherwise, do nothing.

recommend-new-value ()

Warns the user about the inconsistencies in the pair-wise comparisons of the criteria and recommends that the relation between two of the criteria be changed to minimize the inconsistencies.

request-for-menu ()

Prompts the user to select either the Menu or Exit.

restore ()

Restores B to the original B.

restore-ratings ()

Restores the matrix alt-ratings.

scan-row + ()

Puts the indices corresponding to those criteria to which the actual-best alternative is better than the expected-best alternative.

set-menu ()

Prompts the user for the next input. If the next input is 'E', the program will display explanation; 'I', input; 'Q', quit; 'M', menu.

show-greatest-inconsistency ()

Gives a partial list of the inconsistent ratings among criteria. The list is not meant to be exhaustive. It just lists the most inconsistent criteria; other minor inconsistencies are ignored.

sort-crit-list ()

Puts criteria into sorted-crit-list in order of their importance. The most important one is at the top of the list.

stop-scroll ()

Stops scrolling. Scrolling continues when the user presses any key.

sum-of-row ()

Returns the sum of all elements of the j-th row of the array alt-ratings-3.

sum-to-min ()

Uses the input from the Weighted-Sum Rank Method as input to the Min-Max Rank Method.

sum-to-min-menu ()

This menu is activated when the Weighted-Sum Rank Method finishes its execution. The menu asks the user whether he/she wants to use the input from the Weighted-Sum Rank Method as input to the Min-Max Rank Method.

temp-values ()

Computes the two values that optimize the consistency matrix.

unexpected-output ()

Prints the explanation when the expected-best alternative is different from the actual-best alternative.

weighted-sum ()

Drives the weighted-sum rank algorithm.

weighted-sum-pw ()

Implements the Weighted-Sum Pair-Wise Method. It takes the same input format as the Min-Max Pair-Wise Method.

weighted-sum-to-eigen ()

Runs the Min-Max Pair-Wise Method using input from the previous running of the Weighted-Sum Pair-Wise Method.

weighted-sum-to-eigen-menu ()

Is activated when the Weighted-Sum Pair-Wise Method finished its execution. It will ask the user whether he/she wants to use the input from the previous running of the Weighted-Sum Pair-Wise Method to run the Min-Max Pair-Wise Method.

wspw-compute-expected-best ()

Finds the alternative that best satisfies the two most important criteria.

wspw-explain ()

Determines whether expected-best = actual-best and outputs the explanation accordingly.

wspw-handle ()

Performs the appropriate tasks depending on the input x; if x = q, then program quits; x = e, prints explanation; x = i, prints input; x = m, displays menu.

wspw-set-menu ()

Prompts the user for the next input. If the next input is 'E', the program will display explanation; 'I', input; 'Q', quit; 'M', menu.

APPENDIX D:
PROGRAM LISTING FOR
MIN-MAX PAIR-WISE METHOD
(FranzLISP)

```

;-----
;Darlene Teotico
;Naval Ocean Systems Center - Code 444
;-----
(defun fuzzy ()

  ; Create header.
  (princ "                                * A FUZZY LOGIC DECISION-MAKING TOOL
*")(terpri)(terpri)
  (princ "                                a program based on R.Yager's 1977 paper")
  (terpri)
  (princ "                                Multiple Object Decision-Making Using Fuzzy Sets")
  (terpri)

  ; Get labels and ratings for alternatives and criteria.

  (setq alternatives (funcall 'get-labels 'ALTERNATIVE ))
  (setq criteria (funcall 'get-labels 'CRITERION ))
  (setq n (length alternatives))
  (setq m (length criteria))
  (array alt-ratings flonum n m)
  (array B flonum m m)
  (array decision-list fixnum n)
  (get-alt-ratings)
  (get-crit-ratings)

  (compute-eigenvector)
  (find-decision-values)
  (find-best-choice)
]

;-----
; Get labels for criteria and alternatives.

(defun get-labels (type)
  (setq l nil)
  (terpri)
  (terpri)
  (princ "ENTER NUMBER OF ") (princ type) (princ "S: ")
  (setq count (read))
  (terpri)

```

```

(do ((x 1 (add1 x)))
  ((greaterp x count) l)
  (princ " ENTER LABEL FOR ") (princ type) (princ " ")
  (princ x)
  (princ ": ")
  (setq l (append1 l (read))))
]

```

```

;-----
; Menu for degrees of preference.

```

```

(defun alt-ratings-header ()
  (terpri)
  (princ "                RATING THE ALTERNATIVES")
  (terpri)(terpri)
  (princ "                -----") (terpri)
  (princ "                | Enter a value between 0 and 1; |") (terpri)
  (princ "                | where a higher value indicates |") (terpri)
  (princ "                | a better rating.                |") (terpri)
  (princ "                -----") (terpri)
  (terpri) ]

```

```

;-----
; Query user to rate each alternative based on each criterion.

```

```

(defun get-alt-ratings ()
  (do ((x 0 (add1 x)) (xx criteria (cdr xx)))
    ((equal x m) nil)
    (alt-ratings-header)
    (do ((y 0 (add1 y)) (yy alternatives (cdr yy)))
      ((equal y n) nil)
      (princ "For ") (princ (car xx)) (princ ", enter rating for ")
      (princ (car yy)) (princ ": ")
      (store (alt-ratings y x) (read)) )
    (terpri) ]

```

```

;-----
; Header for the RATING THE CRITERIA screen.

```

```

(defun crit-ratings-header ()
  (terpri)

```



```

(princ "By what degree? (Use scale above) ")
(setq degree (read))
(cond ((eq choice 1)
      (store (B i j) (float degree))
      (store (B j i) (quotient 1.0 degree)))
      (t (store (B j i) (float degree))
          (store (B i j) (quotient 1.0 degree)))) ]

;-----
(defun compute-eigenvector ()
  (normalize-B)
  (get-weight-vector)
  (compute-Y-vector)
  (est-lambda-max)

  (setq inconsist
    (sqrt (quotient (diff lambda-max m)(diff (times 2.0 m) 2.0))))
  (cond ((greaterp inconsist .2)
        (terpri)
        (princ "
CRITERIA")(terpri)(terpri)
        (princ " 1")
        (do x 2 (add1 x) (greaterp x m)
          (princ " ") (princ x))
        (terpri) (princ " ")
        (do x 1 (add1 x) (greaterp x m)
          (princ "-----"))
        (princ " ") (terpri)
        (do x 0 (add1 x) (eq x m)
          (princ (add1 x)) (princ " l")
          (do y 0 (add1 y) (eq y m)
            (cond ((lessp (B x y) 1.0)
                  (princ " 1/")
                  (princ (fix (quotient 1 (B x y))))
                  (princ " "))
                  (t(princ " ")
                    (princ (fix(B x y)))
                    (princ " "))))))
          (princ "l") (terpri) )
        (princ " ")
        (do x 1 (add1 x) (greaterp x m)

```

PAIRED-COMPARISON OF

```

        (princ "-----")
        (princ " ") (terpri) (terpri)
        (do ((x 1 (add1 x))
              (c criteria (cdr c)))
            ((null c))
            (princ " ") (princ x) (princ " = ") (princ (car c)) (terpri))
        (terpri)
        (princ " THERE IS INCONSISTENCY IN THE MATRIX OF")
        (princ "FAIRED-COMPARISONS OF CRITERIA") (terpri)(terpri)
        (princ " Lambda max = ") (princ lambda-max) (terpri)
        (princ " Inconsistency = ") (princ inconsist) (terpri)(terpri)
        (princ " Would you like to re-enter the paired-comparison values")
        (terpri)
        (princ " for the criteria? [yes or (no)] ")
        (cond ((eq (read) 'yes)
                (get-crit-ratings)
                (compute-eigenvector))) ]

;-----
(defun normalize-B ()
  (array B-prime flonum m m)
  (do j 0 (add1 j) (eq j m)
    (setq sum 0)
    (do i 0 (add1 i) (eq i m)
      (setq sum (add sum (B i j)))))
  (do i 0 (add1 i) (eq i m)
    (store (B-prime i j) (quotient (B i j) sum)))) ]

;-----
(defun get-weight-vector ()
  (setq W nil)
  (do i 0 (add1 i) (eq i m)
    (setq sum 0.0)
    (do j 0 (add1 j) (eq j m)
      (setq sum (add sum (B-prime i j)))))
  (setq W (append1 W (quotient sum (float m))))) ]

;-----
(defun compute-Y-vector ()
  (setq Y nil)
  (do i 0 (add1 i) (eq i m)

```

```

(setq sum 0.0)
(do j 0 (add1 j) (eq j m)
  (setq sum (add sum (times (B i j) (nth j W))))))
(setq sum (quotient sum (nth i W)))
(setq Y (append1 Y sum)) ]

```

```

;-----
(defun est-lambda-max ()
  (setq sum 0.0)
  (do i 0 (add1 i) (eq i m)
    (setq sum (add sum (nth i Y))) )
  (setq lambda-max (quotient sum (float m))) ]

```

```

;-----
(defun find-decision-values ()
  (setq decision-list nil)
  (do A 0 (add1 A) (eq A n)
    (setq min 99999.0)
    (do C 0 (add1 C) (eq C m)
      (setq new (expt (alt-ratings A C)
        (times (nth C W)(float m))))
      (cond ((lessp new min)
        (setq min new))))
    (setq decision-list (append1 decision-list min))) ]

```

```

;-----
(defun find-best-choice ()
  (setq value -1.0)
  (do A 0 (add1 A) (eq A n)
    (cond ((greaterp (nth A decision-list) value)
      (setq choice A)
      (setq value (nth A decision-list)))))
  (terpri)
  (princ "Based on the model, the best choice is ")
  (princ (nth choice alternatives)) (terpri) ]

```

APPENDIX E:
PROGRAM LISTING FOR
MIN-MAX RANK METHOD
(FranzLISP)

```

;-----
;-----
;This fuzzy logic decision tool is based on Ronald R. Yager's
;"A New Methodology For Ordinal Multiobjective Decisions Based On
;Fuzzy Sets"
;
;Darlene Teotico
;Naval Ocean Systems Center - Code 444
;-----
(defun fuzzy ()

  ; Create header.
  (terpri)
  (princ "
                                * A FUZZY LOGIC DECISION-MAKING TOOL
*")(terpri)(terpri)
  (princ "
                                a program based on R.Yager's 1981 paper")
  (terpri)
  (princ "
                                A New Methodology for Ordinal Multiobjective")
  (terpri)
  (princ "
                                Decisions Based on Fuzzy Sets")
  (terpri)
  ; Get labels and ratings for alternatives and criteria.

  (setq alternatives (funcall 'get-labels 'ALTERNATIVE ))
  (setq criteria (funcall 'get-labels 'CRITERION ))
  (setq no-alts (length alternatives))
  (setq no-crits (length criteria))
  (array alt-ratings fixnum no-crits no-alts)
  (array crit-ratings fixnum no-crits)
  (array decision-list fixnum no-alts)
  (array c-min fixnum no-alts)
  (get-alt-ratings)
  (get-crit-ratings)

  (adjust-ratings)
  (setq decision (make-decision))
  (terpri) (terpri) (princ "Our decision based on this model is ")
  (princ (nth (cdr decision) alternatives)) (terpri)
]

```

; Get labels for criteria and alternatives.

```
(defun get-labels (type)
  (setq l nil)
  (terpri)
  (terpri)
  (princ "ENTER NUMBER OF ") (princ type) (princ "S: ")
  (setq count (read))
  (terpri)
  (do ((x 1 (add1 x)))
      ((greaterp x count) l)
    (princ " ENTER LABEL FOR ") (princ type) (princ " ")
    (princ x)
    (princ ": ")
    (setq l (append1 l (read))))
  )
```

; Preference measures menu.

```
(defun print-measures ()
  (terpri)
  (princ "
  ----- ") (terpri)
  (princ " | PREFERENCE MEASURES: |") (terpri)
  (princ " | 0. lowest or none |") (terpri)
  (princ " | 1. very low |") (terpri)
  (princ " | 2. low |") (terpri)
  (princ " | 3. medium |") (terpri)
  (princ " | 4. high |") (terpri)
  (princ " | 5. very high |") (terpri)
  (princ " | 6. perfect or absolute |") (terpri)
  (princ "
  ----- ") (terpri)
  )
```

; Query user to rate each alternative based on each criterion.

```
(defun get-alt-ratings ()
  (print-measures)
  (do ((x 0 (add1 x)) (xx criteria (cdr xx)))
```

```

(equal x no-crits) nil)
(do ((y 0 (add1 y)) (yy alternatives (cdr yy))))
  ((equal y no-alts) nil)
  (princ "For ") (princ (car xx)) (princ ", enter rating for ")
  (princ (car yy)) (princ " (0 - 6): ")
  (store (alt-ratings x y) (read)) ]

```

```

; Query user to rate each criterion. Then adjust the rating that is the
; negation of the value.

```

```

(defun get-crit-ratings ()
  (print-measures)
  (do ((x 0 (add1 x)) (xx criteria (cdr xx)))
    ((equal x no-crits) nil)
    (princ "Enter rating for ") (princ (car xx))
    (princ " (0 - 6): ")
    (store (crit-ratings x) (difference 6 (read)))) ]

```

```

; Adjust the ratings of the alternatives using  $C_i = b_i \vee A_i$ .
; where  $C_i$  is the new rating,  $b_i$  is the adjusted criterion rating and
;  $A_i$  is the actual alternative rating.

```

```

(defun adjust-ratings ()
  (cond ((null (car criteria)) criteria)
        (t (do x 0 (add1 x) (equal x no-crits)
              (do y 0 (add1 y) (equal y no-alts)
                (cond ((greaterp (crit-ratings x)
                                   (alt-ratings x y))
                      (store (alt-ratings x y) (crit-ratings x)))))) ]

```

```

; Create the set  $D = C_1 \wedge C_2 \wedge \dots \wedge C_p$ , where  $D(x) = \text{Min}[C_i(x)]$ 
; and  $C_i(x) = B_i \vee A_i(x)$ .

```

```

(defun make-decision-list (alt-list decision-list)
  (do a alt-list (cdr a) (null a)
    (setq max-alt (cons (alt-ratings 0 (car a)) (car a)))
    (store (c-min (car a)) 0)

```

```

(do c 1 (add1 c) (equal c no-crits)
  (cond ((lessp (alt-ratings c (car a))
    (car max-alt))
    (setq max-alt (cons (alt-ratings c (car a)) (car a)))
    (store (c-min (car a)) c))) )
(setq decision-list (append1 decision-list max-alt)))

; Sort the decision list
(setq decision-list (sortcar decision-list 'greaterp)) )

;-----
; Find the optimal solution  $x^*$  that satisfies  $D(x^*) = \text{Max } D(x)$  for
; all  $x$  where  $x$  is an element of the set of alternatives.

(defun make-decision ()
  (prog ()
    ; create the list of alternatives...
    (setq alt-list nil)
    (do a 0 (add1 a) (eq a no-alts)
      (setq alt-list (append1 alt-list a)))
    (setq decision-list nil)
    (setq decision-list (make-decision-list alt-list decision-list))
    (setq decision (car decision-list)) ; Make max be the first in list.
    (setq decision-list2 decision-list) ; make a working copy of d-list.

  FIND-TIES
  (setq alt-list (cons (cdr decision) nil))
  (do d (cdr decision-list2) (cdr d) (null d)
    (cond ((eq (caar d) (car decision))
      (setq alt-list (append1 alt-list (cdar d)))
      (do c 0 (add1 c) (eq c no-crits)
        (cond ((eq (car decision)
          (alt-ratings c (cdar d))
          (store (alt-ratings c (cdar d)) 7)))))))
    (cond ((greaterp (length alt-list) 1)
      (do c 0 (add1 c) (eq c no-crits)
        (cond ((eq (car decision)
          (alt-ratings c (cdr decision))
          (store (alt-ratings c (cdr decision)) 7))))
      (setq decision-list2 (make-decision-list alt-list decision-list2))
      (setq decision (car decision-list2))

```

```
(cond ((less-than-7 alt-list) (go FIND-TIES))))))
```

```
(return decision)
```

```
]
```

```
;-----  
; Checks for ratings less than 7. If there are none, then there will be  
; a tie...
```

```
(defun less-than-7 (alt-list)  
  (setq below-7 nil)  
  (do ((a alt-list (cdr a))  
      (below-7 nil))  
      ((or (null a) below-7) below-7)  
    (do c 0 (add1 c) (eq c no-crits)  
      (cond ((lessp (alt-ratings c (car a)) 7)  
            (setq below-7 t)) ]
```